Baze de date pentru începători

John V. Petersen

Cuprins

Index

A

B

C

D

G-H

I

U-V

X-Z

Despre autor

John V. Petersen este președintele și fondatorul firmei de aplicații software și design pentru baze de date Main Line Software, Inc., cu sediul în Pennsylvania. Platformele pe care lucrează MLSI sunt Visual FoxPro, Visual Basic, Access, SQL Server și noua platformă de la Microsoft, Visual Studio.NET.

John a obținut un MBA la Haub School of Business în cadrul St. Joseph's University în 1993. În prezent, John este 2L la Rutgers University School of Law. John a susținut conferințe la multe evenimente din domeniu, printre care Microsoft Developer Days și Microsoft Tech-Ed. Este autorul mai multor cărți și articole în domeniu și al mai multor dezvoltări de aplicații și proiecte de baze de date. El a primit în anul 1996 premiul Most Valuable Professional (cel mai valoros profesionist), acordat de Microsoft.

Dedicație

Fiilor mei, Karl și Keith. Sunt mândru de tot ceea ce faceți și știu că veți realiza amândoi lucruri grozave. Vă iubesc foarte mult. Soției mele, Evelyn. Îți mulțumesc din nou pentru toată dragostea și tot sprijinul pe care mi le-ai dăruit de-a lungul anilor, mai ales acum, în perspectiva unui nou început al meu în cariera juridică.

Mulțumiri

Îi mulțumesc în primul rând lui Jenny Watson pentru că mi-a propus acest proiect. Inițial, acesta a avut o mulțime de începuturi și opriri din cauza programului meu încărcat. Jenny a crezut în două lucruri: în proiect și în faptul că îl voi realiza eu. Îți mulțumesc, Jenny, pentru răbdare și ajutor!

Un autor este foarte norocos când lucrează cu un editor de dezvoltare precum Maureen McDaniel. Ori de câte ori am avut o întrebare sau mă preocupa ceva, Maureen a fost mereu aproape, gata să-mi dea răspunsul corect și îndrumările necesare. Îți mulțumesc, Maureen, pentru că datorită ție acest proiect a devenit amzant și plin de satisfacții.

Am avut și privilegiul să mă bucur de prietenia și ajutorul lui Jim Booth și am fost încântat că a acceptat să fie editorul tehnic al acestei cărți. Jim este unul dintre cei mai inteligenți oameni pe care îi cunosc, iar cartea a beneficiat din plin de experiența, înțelepciunea și ajutorul lui.

A scrie o carte este un lucru important. Necesită mult timp și, într-o anumită măsură, sacrificii. În timp ce am scris această carte, în afară

de faptul că trebuia să țin pasul cu cei doi fii ai mei foarte activi, să-mi văd soția când și când și să conduc o firmă de consultanță software, am utilizat puținul timp liber pe care îl aveam pentru a studia dreptul la Rutgers University School of Law – camden. Pentru că am putut să țin pasul cu aceste obligații, trebuie să mulțumesc: mai întâi colegei mele de facultate Stephanie Meade, fiindcă m-a tot bătut la cap să-mi îndeplinesc îndatoririle legate de facultate, apoi profesoarei Beth Hillman, fiindcă a „suflat viață" asupra Constituției SUA și mi-a deschis ochii asupra principiilor pe care le apără SUA prin sistemul lor legislativ. Încurajările voastre au făcut ca întoarcerile mele de la facultate la ora 22:30 să fie mult mai suportabile. În al treilea rând, trebuie să mulțumesc profesorului meu de corespondență oficială, Brian Foley, pentru că datorită lui am devenit un scriitor mai bun. În sfârșit, îi mulțumesc celui mai bun prieten al meu, Rod Paddock. Nu pot exprima prin cuvinte ce mult au însemnat prietenia și sprijinul tău de-a lungul anilor. Rămânând fidel filozofiei tale, „mai puțin înseamnă mai mult", cred că asta e tot ce trebuie să spun.

Spuneți-ne părerea dumneavoastră!

În calitate de cititor al cărții, *dumneavoastră* sunteți cel mai important critic și comentator al ei. Vă prețuim opiniile și dorim să știm ce e bine, ce ați dori să mai publicăm, cu alte cuvinte dorim să ne împărtășiți cât de poate de mult din experiența dumneavoastră.

Ca editor asociat la Que, mă bucur de orice comentarii. Puteți expedia un fax, un e-mail sau îmi puteți scrie chiar mie pentru a mă informa ce v-a plăcut și ce nu la această carte – precum și ce aș putea face pentru a o îmbunătăți.

Vă rog să nu uitați că nu vă pot ajuta în privința problemelor tehnice legate de subiectul cărții, și asta din cauza volumului imens de mesaje pe care le primesc, motiv care mă face și să nu pot răspunde tuturor acestor mesaje.

Atunci când îmi scrieți, nu uitați să numele și autorul cărții la care vă referiți, dar și numele, adresa și telefonul sau faxul dumneavoastră. Voi citi cu atenție toate comentariile și le voi face cunoscute autorilor și editorilor care au lucrat la cărțile respective.

Fax: 317-581-4666
E-mail: feedback@quepublishing.com
Adresa: Dean Miller
Editor asociat
Que
201 West 103rd Street
Indianapolis, IN 46290 USA

INTRODUCERE

Credeți sau nu, piața nu oferă prea mult unei persoane care dorește o carte fundamentală și ușor de citit despre baze de date. Când mi-a fost propus acest proiect mi-am spus: „Trebuie să existe o mulțime de cărți pe piață care explică noțiunile fundamentale ale bazelor de date pentru începători." Spre surprinderea mea, am găsit foarte puține și atunci mi-am dat seama că această carte trebuie scrisă.

Partea cea mai grea în orice muncă este să începi. După ce ai pornit pe un drum, și, poate, cel mai important lucru este să fie drumul cel bun, restul este ușor. Scopul acestei cărți este de a vă orienta în direcția corectă când vă începeți drumul în lumea tehnologiei bazelor de date.

Dorința autorului a fost să realizeze o carte agnostică, nu în mod special destinată unui program anume pentru baze de date. Deși exemplele date se folosesc de Access, conceptele bazelor de date prezentate se aplică oricărei baze de date relaționale. De ce am ales Access ca platformă pentru prezentarea conceptelor? Din două motive. În primul rând, Access este de departe sistemul penru baze de date personale cel mai ușor de utilizat și învățat. Apoi sunt șanse foarte mari să aveți Microsoft Office. În consecință, probabil că dețineți deja o versiune de Microsoft Access. Deși Access 2000 a fost folosit pentru realizarea acestei cărți, alte versiuni (Access 97, sau ultima versiune, Access XP) ar trebui să funcționeze la fel de bine.

Cui i se adresează cartea?

Această carte le este destinată celor care au auzit de baze de date dar nu știu ce sunt acestea, ce conțin sau cum să le construiască. Este scrisă pentru a ajuta orice persoană care trebuie să lucreze prima oară cu o bază de date. Această necesitate poate fi o obligație de serviciu sau o dorință personală de îmbunătățire a nivelului de aptitudini și cunoștințe. Indiferent de modul în care ați ajuns pe punctul să vă începeți drumul în acest domeniu, dacă sunteți începător trebuie să porniți cu pasul 0, iar această carte este exact ce vă trebuie!

Cum este organizată cartea

Cartea este alcătuită din cinci părți. Această structură vă poartă de-a lungul evoluției conceptelor, analizei, proiectării, construirii și, în final, implementării bazelor de date.

Partea I – Baze de date – elemente generale

Înainte de a merge pe două picioare, oricine merge de-a bușilea. Ce este o bază de date? Care sunt tipurile de baze de date și cum sunt ele utilizate azi în afaceri? Aceste aspecte și multe altele sunt explorate în partea I.

Partea a II-a – Proiectarea și construirea primei baze de date

După ce am prezentat aspectele teoretice legate de bazele de date, este timpul să trecem la treabă. Primul pas în construirea unei baze de date constă în analiza unor probleme de afaceri și proiectarea unei baze de date pentru a le rezolva. Partea a II-a vă călăuzește pașii de la designul conceptual la construirea efectivă a unei baze da date.

Partea a III-a – Întreținerea bazei de date

După ce ați construit baza de date, ea trebuie întreținută. Ce anume menține o bază de date în bună stare de funcționare? O bază de date este la fel de bună ca și datele pe care le conține. Partea a III-a tratează conceptele cu ajutorul cărora vă puteți asigura că în baza de date sunt stocate numai date valide.

Partea a IV-a – Extragerea informațiilor din baza de date – introducere în SQL

În acest punct știți ce este o bază de date, cum să proiectați, să construiți și să întrețineți o bază de date. Partea a IV-a revine la scopul inițial: punerea la lucru a unei baze de date. Totul începe de la modul în care sunt extrase informațiile din ea. Această carte explică limbajul bazelor de date: SQL (Structured Query Language).

Partea a V-a – Funcționarea bazei de date – construirea unei aplicații simple pentru baze de date în Access

Partea a V-a pune lucrurile cap la cap, arătându-vă cum să prezentați baza de date utilizatorilor finali. Într-un anumit moment, oamenii vor trebui să interacționeze cu baza dumneavoastră de date. Această parte tratează aspecte precum formularele pentru introducerea datelor și rapoartele în contextul unei aplicații simple pentru baze de date din Access.

Convenții utilizate în această carte

Structura este concepută pentru a vă ajuta să înțelegeți și să vă însușiți cât mai multe informații. La începutul fiecărui capitol veți găsi un rezumat, sub forma unei liste conținând subiectele tratate în el.

În plus, veți găsi mai multe pictograme de-a lungul cărții. Ele vă atrag atenția spre informații mai detaliate, sfaturi spre a vă face munca mai productivă sau atenționări pentru a vă ajuta să evitați potențialele probleme. Iată care sunt aceste pictograme:

Notă

Notele oferă informații suplimentare despre subiectul tratat într-o anumită secțiune. Puteți sări peste ele, învățând numai conceptele cheie prezentate.

Indiciu

Indiciile vă oferă o imagine asupra „a ceea ce se întâmplă aici". Învățând o parte din informațiile de fundal despre o anumită sarcină, veți înțelege mai bine conceptul prezentat. Indiciile vor face și ca eforturile dumneavoastră de dezvoltare să fie mai ușoare și mai eficiente.

Atenție

Atenționările sunt menite să salveze situația! În orice mediu de dezvoltare a aplicațiilor sunt 100 de lucruri care ar putea merge prost. Mai ales în situațiile în care ați putea ajunge să munciți foarte mult și să vă îndepărtați de „calea cea dreaptă", veți găsi o atenționare care vă ajută să evitați acest lucru.

PARTEA I

BAZE DE DATE – ELEMENTE GENERALE



1

INTRODUCERE ÎN BAZELE DE DATE

În acest capitol:

- Ce este o bază de date?
- Cum sunt utilizate bazele de date
- Sisteme tipice de management al bazelor de date utilizate azi
- Termeni cheie în bazele de date
- Modelul relațional: prezentare generală
- Profesiuni în domeniu
- Rezumat

„Drumul de 1000 de li începe cu primul li."

– Vechi proverb chinez

Capitolul tratează elementele esențiale ale bazelor de date relaționale, modul lor de utilizare și termenii cheie. Există mai multe tipuri de baze de date, precum rețea, ierarhice și orientate obiect. Subiectul de interes în această carte îl reprezintă bazele de date relaționale, care sunt, de departe, tipul cel mai popular de baze de date utilizat astăzi. Caracteristicile acestui tip de baze de date sunt prezentate în decursul acestui capitol.

Când începeți studiul bazelor de date s-ar putea să aflați că, deși tehnologia a făcut bazele de date mai puternice ca oricând, în esență, acestea există de mai mult timp decât computerele. Acest lucru s-ar putea să vă surprindă, dar după ce veți înțelege ce sunt de fapt bazele de date, veți înțelege și de ce lucrurile stau astfel. Întrucât ați ales această carte, voi presupune că studiul bazelor de date este ceva nou pentru dumneavoastră și că studiul bazelor de date nu vă este familiar. Cel mai bun sfat pe care vi-l pot oferi în acest moment este să vă legați centura de siguranță, să respirați adânc, să fiți pregătit să vă distrați și să învățați multe lucruri noi și utile!

Ce este o bază de date?

Cele mai bune definiții sunt acelea care necesită cât mai puține cuvinte. În cazul bazelor de date, definiția cea mai concisă poate fi redusă la un singur cuvânt: ***colecție***. ***O bază de date*** este o colecție de date corelate. Cele trei elemente esențiale ale unei baze de date sunt:

- Tabelele
- Coloanele
- Liniile (rândurile)

Tabelele, numite și ***entități***, reprezintă elementele de bază ale informațiilor pe care doriți să le urmăriți. Așa sunt, de exemplu, clienții unei firme de servicii, comenzile, facturile produselor vândute. Rezultă, deci, că o bază de date tipică pentru o firmă conține tabele pentru clienți, comenzi, facturi.

Coloanele, numite și ***câmpuri***, reprezintă atributele unui tabel. Considerați tabelul cu clienții unei firme. Cum ați descrie o entitate client? Atribute precum numele prenumele, adresa, orașul, codul poștal, telefonul etc. sunt informații care trebuie stocate.

Liniile (rândurile), numite și ***înregistrări***, reprezintă datele propriu-zise. În timp ce câmpurile descriu ce tip de date sunt stocate, în liniile tabelei sunt stocate datele.

Sistemele moderne de management al bazelor de date administrează ansamblul de tabele, linii și coloane. Cuvântul ***moderne*** din propoziția anterioară sugerează că au existat sau există baze de date nemoderne? Răspunsul este afirmativ. Gândiți-vă la bibliorafturile cu facturi din departamentul de contabilitate a firmei dumneavoastră. Acestea pot fi considerate ca niște

tabele, în care câmpurile de date din factura obișnuită pot fi considerate coloane, iar fiecare factură poate fi consideată un rând al tabelului. De asemenea, gândiți-vă la vechiul sistem cu fișe de catalog pe care bibliotecile îl utilizau (în România este încă utilizat în multe biblioteci –n.t.). Fiecare fișă conține aceleași elemente: titlul cărții, autorul, anul apariției, editura etc. Este ușor de văzut că fișele formează tabelul, elementele conținute de fiacare fișă sunt coloanele și fiecare fișă este o linie de date.

Diferența esențială între bazele de date moderne și „strămoșii" lor bazați pe documente este posibilitatea de regăsire a datelor. Într-un sistem modern de baze de date, găsirea tuturor cărților publicate în anul 2000 este simplă: scrieți o interogare, punând computerului o întrebare. Sunt șanse să primiți răspunsul în mai puțin de o secundă. În sistemele vechi, bazate pe documente, acest lucru ar fi durat câteva minute, câteva ore sau poate câteva zile.

În esență, tabelele, coloanele și liniile sunt „cărămizile" pentru o bază de date relațională. Atunci când construiți o bază de date, tabelele, liniile și coloanele intervin imediat în discuție. O bază de date pentru o firmă poate conține peste 100 de tabele. Fiecare tabel poate avea unul sau 100 de câmpuri. Numărul optim al acestora poate fi determinat prin tehnici corecte de design, care sunt discutate în această carte. Deși bazele de date sunt interesante, modul lor de utilizare este, poate, și mai interesant.

Utilizarea bazelor de date

Dacă sunteți ca majoritatea oamenilor, și dumneavoastră primiți cantități importante de corespondență nesolicitată (***junk mail***). Se pare că imediat ce oferiți cuiva informații despre dumneavoastră, începeți să primiți astfel de mesaje – fie pe căile tradiționale, fie pe căi electronice. Dacă ați cumpărat ceva de la firma X, veți primi informații și despre firma Y, care oferă produse și servicii asemănătoare. Întrebarea dumneavoastră ar putea fi: „De unde au știut să-mi trimită aceste informații?" sau „De unde au știut că am cumpărat produsul respectiv?".

Răspunsul are două aspecte. Mai întâi, există informații despre dumneavoastră pe care le-ați furnizat direct sau indirect. Apoi, firmele au asimilat aceste informații în baze de date. Oricând cumpărați ceva folosind o carte de credit, oferiți cuiva informații despre dumneavoastră și despre oamenii ca dumneavoastră. Dumneavoastră primiți lunar un raport cuprinzând cumpărăturile lunii încheiate. Datele privind aceste cumpărături, precum și cele ale altor milioane de indivizi care folosesc cărți de credit sunt stocate în baze de date.

Se poate spune că la baza construcției bazelor de date stă transformarea datelor disparate în informații. În lumea afacerilor de azi, și mai ales în marketing, scopul bazelor de date este și el dublu. Mai întâi, firmele doresc să știe despre dumneavoastră cât mai mult posibil. Apoi, în acest fel află ce anume vă doriți. Și astfel revenim la exemplul anterior cu cartea de credit.

În acest proces, firmele își pot direcționa eforturile de marketing. Care este rezultatul final? Ca și cosumator, este mai probabil să cumpărați decât să vindeți. Și cui trebuie să mulțumească firmele pentru creșterea vânzărilor? Bazelor de date, desigur!

Ați cumpărat vreodată ceva de pe Amazon.com? La început, datele au fost utilizate incidental pe site-ul Amazon. Totuși, cu trecerea timpului, o dată ce s-au adunat datele și experiența, datele au devenit o parte integrantă a site-ului. Azi, când intrați pe acest site – presupunând, desigur, că v-ați abonat – vă sunt oferite tot felul de sugestii privind produsele. Amazon stochează datele oferite de cumpărăturile pe care le-ați făcut într-o bază de date și le utilizează pentru a anticipa ce alte produse ați fi interesat să cumpărați.

Valoarea predictivă a datelor este, probabil, beneficiul cel mai important al stocării datelor în baze de date. Companiile folosesc bazele de date pentru a afla evoluția afacerilor și a anticipa traiectoria lor viitoare. Analizând datele, companiile pot lua măsuri corective dacă sunt detectate probleme.

Aspectele prezentate anterior nu reprezintă deloc o listă exhaustivă a modurilor de utilizare a unei baze de date. Nu ați văzut decât partea aisbergului de deasupra apei. Zilnic sunt descoperite noi căi de utilizare a datelor. Prin tehnologia modernă, bazele de date vă pun la dispoziție milioane de bytes (octeți) simultan sub formă de informații. Dacă datele ar trebui să fie compilate manual, ar fi nevoie de ore, zile, săptămâni, luni sau poate chiar ani! Acest beneficiu este extrem de important, deoarece, atunci când datele ating o anumită vârstă, valoarea lor devine egală cu zero.

Multe sisteme de administrare a bazelor de date sunt folosite azi. Secțiunea următoare le descrie pe cele mai populare.

Sisteme tipice de administrare a bazelor de date

Multe sisteme de administrare a bazelor de date sunt folosite azi. Printre cele mai populare sunt:

- Microsoft Access
- Microsoft SQL Server
- MSDE (Microsoft Data Engine)
- Oracle
- IBM DB2

Microsoft Access este unul dintre cele două programe importante pentru baze de date oferite de Microsoft. Access este o platformă pentru baze de date orientată spre rețele și birou. Deși are foarte multe funcții, este limitată atât de cantitatea de date pe care le poate stoca, dar și de numărul de utilizatori pe care îi poate servi simultan. În general, dimensiunea maximă a unei baze de date în Access este de 2 milioane de bytes (2 gigabytes). Marele avantaj pe care îl oferă Access este ușurința de utilizare. Acesta este motivul pentru care am ales Access pentru a ilustra exemplele din carte.

Pe de altă parte, există produse precum Microsoft SQL Server, Oracle și DB2. Aceste sisteme de administrare a bazelor de date pot stoca miliarde de bytes (terabytes). Mai mult, aceste sisteme pot servi mii de utilizatori simultan. Cele mai mari site-uri și aplicații pentru baze de date folosesc produse de acest tip. Deoarece această carte își propune să fie o introducere în bazele de date, produsele precum SQL Server și Oracle îi depășesc scopul. Citiți anexele, pentru informații despre resurse de studiu mai avansate. Este important să rețineți că, deși cartea folosește Microsoft Access pentru a ilustra funcționarea bazelor de date, dumneavoastră veți cunoaște toate funcțiile acestora. Scopul acestei cărți este de a vă învăța ce sunt bazele de date și cum să le folosiți. În esență, toate bazele de date funcționează după aceleași principii teoretice. După ce veți cunoaște elementele de bază prezentate în carte, vă va fi relativ ușor să faceți transferul de cunoștințe la platforme mai avansate, precum SQL Server, Oracle, sau DB2.

Până acum, aceste platforme au fost denumite sisteme de administrare a bezelor de date (***database management system – DBMS***). Mai precis ele sunt sisteme de administrare a bazelor de date relaționale. Construcția tuturor sistemelor menționate are la bază modelul relațional. Înțelegerea bazelor de date are la bază două etape importante: cunoașterea câtorva termeni de bază și înțelegerea principiilor cheie ale modelului relațional. Aceste etape fac obiectul următoarelor două secțiuni.

Termeni cheie în bazele de date

Un glosar complet cu termeni din bazele de date poate fi găsit în anexa C a acestei cărți. Cu riscul redundanței, este cel mai bine să vă familiarizați cu cei mai obișnuiți și mai importanți termeni pe care nu îi veți întâlni numai în această carte, ci în orice activitate care are legătură cu bazele de date pe care o veți desfășura.

Bază de date

O bază de date a fost definită anterior în acest capitol ca fiind o colecție de elemente înrudite de date. Mai precis, o bază de date este o colecție de tabele sau entități înrudite de date. De exemplu o bază de date obișnuită pentru o firmă ar putea fi alcătuită din tabelele: client, comanda, produsul comandat. Toate aceste tabele sunt înrudite unul cu altul într-un anumit mod. În acest exemplu, clienții dau comenzi, iar comenzile conțin produse. Chiar dacă fiecare tabel este separat, ele alcătuiesc împreună baza de date.

Tabele și entități

Un tabel reprezintă „ceva" despre firmă. Tabelele sunt numite și entități. Dacă doriți să găsiți sau să stocați informații despre ceva, sunt șanse ca acel ceva să fi fost definit ca tabel. Unul dintre primii pași în construirea unei baze de date presupune definirea tabelelor din care este formată.

Câmpuri, coloane și atribute

Așa cum o bază de date este alcătuită din tabele, tabelele sunt formate din câmpuri. Câmpurile sunt numite ***coloane*** sau ***atribute***. În funție de context, va fi folosit unul sau altul dintre acești termeni. Nu vă lăsați derutat de folosirea mai multor cuvinte, deoarece câmp, coloană și atribut înseamnă toate același lucru. O bună metodă de a înțelege ce sunt câmpurile este să vă gândiți la ele ca descriptori ai tabelelor. De exemplu, cum ar putea fi descrisă entitatea client? Exemple potrivite de câmpuri ar fi numele de familie, prenumele, orașul, statul, codul poștal, numărul de telefon etc.

Normalizarea

Când creați o bază de date, definirea tabelelor și a câmpurilor nu este suficientă. Numărul și structura optimă a tabelelor sunt determinate printr-un proces numit ***normalizare***. Procesul de normalizare presupune crearea a numeroase tabele mici în loc de a crea mai puține tabele monolitice. Unul dintre scopurile principale ale normalizării este reducerea redundanței datelor. Subiectul normalizării este discutat în capitolul 4.

Rânduri și înregistrări

Fiecare tabel dintr-o bază de date conține zero sau mai multe rânduri. Rândurile sunt numite și ***înregistrări***. De exemplu, fiecare client al unei firme are o înregistrare separată în tabelul cu date despre clienți. De aceea fiecare comandă a unui anumit client reprezintă un rând în tabelul care conține date despre comenzi.

Cheie primară

O expresie care identifică unic o înregistrare dintre toate celelalte se numește ***cheie primară***. Subiectul cheilor și indexurilor este tratat în capitolul 3.

Modelul bazei de date relaționale: prezentare generală

În anul 1970, dr. E. F. Codd, cercetător la IBM, a publicat un articol intitulat ***A Relational Model of Data for Large Shared Data Banks*** (un model relațional de date pentru bazele de date partajate de mulți utilizatori). În acest articol, dr. Codd a propus o nouă teorie a reprezentării bazelor de date a cărei natură era în mod esențial matematică. Teoria a propus și noi modele matematice pentru accesarea și manipularea unei baze de date. Recunoscând unele dintre dificultățile care pot fi întâlnite, autorul a formulat regulile pentru normalizarea unei baze de date. Aceste reguli sunt discutate în capitolul 4.

În secțiunea anterioară au fost amintite câteva sisteme „relaționale" de baze de date disponibile azi pe piață. Folosirea cuvântului relațional într-un astfel de context l-a deranjat pe autorul articolului. Preocuparea lui era confuzia existentă

între ideea că datele ar putea fi reprezentate în tabele înrudite și teoria sa privind bazele de date relaționale. De aceea, el a creat în anul 1988 cele 12 reguli faimoase ale teoriei bazelor de date relaționale. Cele 12 reguli sunt cuprinse în lista următoare: atât ca formulare, dar și ca explicare a înțelesului lor. Unele concepte cu care nu sunteți încă familiarizat ar putea fi menționate în aceste reguli și explicații. Unde este posibil, sunt indicate și trimiteri la locurile unde sunt tratate aceste subiecte în carte. Iată aceste reguli:

1. Toate informațiile dintr-o bază de date relațională sunt reprezentate explicit la nivel logic și exact într-un singur mod – prin valori în tabele.

Indiciu

Toate datele dintr-o bază de date sunt conținute în tabele. Valorile din tabele sunt reprezentate sub formă de coloane (câmpuri).

2. Fiecare element de date (valoare atomică) dintr-o bază de date relațională este logic accesibil ca rezultat al unei combinații între numele tabelului, cheia primară și numele coloanei.

Indiciu

Datele dintr-un tabel nu sunt sub forma unui șir continuu de date, ci datele pot fi accesate precizând coloana din care fac parte. De exemplu, puteți referi numărul comenzii din tabelul cu comenzi ca orders.ordernumber. cu alte cuvinte, nu este necesar să căutați într-un șir de caractere pentru a localiza numărul comenzii.

3. Valorile nule (diferite de spațiul necompletat sau de un șir de caractere blanc) sunt acceptate într-un DBMS relațional, pentru a reprezenta informațiile lipsă și situațiile în care nu pot fi date informații în mod sistematic, indiferent de tipul datelor.

Indiciu

Valorile nule pot fi la început un concept greu de înțeles. Gândiți-vă la un câmp gol într-o înregistrare pentru un client. Câmpul este necompletat, sau valoarea sa nu este cunoscută? Valorile nule indică o situație în care valoarea este necunoscută, spre deosebire de cazul în care câmpul este necompletat. Gândiți-vă la valorile nule ca având semnificația „Nu știu".

4. Descrierea bazei de date este reprezentată la nivel logic în același mod ca pentru datele obișnuite, deci utilizatorii autorizați pot să aplice același limbaj relațional pentru a determina structurile de date care se aplică pentru datele obișnuite.

Indiciu

În procesul construirii unei baze de date, veți defini elemente ca tabele, câmpuri, indexuri și așa mai departe. Numele tabelelor, câmpurilor, indexurilor etc. sunt date care descriu alte date, numite ***metadate***. În lumea bazelor de date, metadatele sunt numite și dicționare de date. Dicționarele de date pentru adevăratele baze de date relaționale sunt stocate în tabele de date similare tabelelor de date pe care le definiți. Mai mult, același limbaj pe care îl folosiți la extragerea datelor din tabelele pe care le proiectați poate fi folosit în tabelele pe care sistemul le definește pentru stocarea dicționarelor de date.

5. Un sistem relațional acceptă mai multe limbaje și diferite moduri de utilizare la terminal. Totuși, trebuie să existe cel puțin un limbaj în care să fie exprimate în declarații, cu o sintaxă bine definită, șirurile de caractere și care este comprehensiv în acceptarea următoarelor:
   - Definiții ale datelor
   - Definiții ale vizualizărilor
   - Manipularea datelor
   - Condiții (restricții) de integritate
   - Autorizări

Indiciu

Această regulă se referă la capacitatea unei baze de date de a suporta limbaje care vă permit să lucrați, să vedeți, să manipulați, să controlați accesul și să protejați datele. Deși nu este declarat clar, cel mai popular limbaj pentru a realiza toate aceste sarcini este SQL (Structured Query Language). Detalii despre ce este SQL și cum poate fi implementat veți găsi în partea a patra a cărții.

6. Toate vizualizările care, în teorie, sunt actualizabile pot fi actualizate de sistem.

Indiciu

Această regulă înseamnă că un DBMS relațional are încorporată capacitatea de a modifica datele. În produse precum SQL Server și Oracle pot fi stabilite sarcini precum actualizarea datelor și întreținerea bazei de date la anumite intervale de timp.

7. Capacitatea de administrare a unei relații de bază sau a uneia derivate ca un singur operand se aplică nu numai pentru regăsirea datelor, dar și pentru introducerea, actualizarea și ștergerea datelor.

Indiciu

O ***vizualizare*** poate fi considerată ca o fereastră prin care pot fi privite datele și se poate lucra cu ele. Ea poate fi formată din coloane ale unuia sau mai multor tabele. Deoarece o vizualizare este o entitate unică, alcătuită din unul sau mai multe tabele, este necesară numai o operație pentru a regăsi, introduce, actualiza sau șterge date din unul sau mai multe tabele simultan.

8. Programele pentru aplicații și activitățile de la terminal rămân separate logic oricând se produc modificări în reprezentările de stocare sau metodele de acces.

Indiciu

Utilizatorul nu trebuie să aibă cunoștințe de specialitate în legătură cu locul de stocare fizică a datelor sau cu platforma pe care acestea sunt stocate. Cu alte cuvinte, utilizatorul nu are nevoie să știe dacă datele sunt stocate pe Unix sau pe Windows 2000 Server. În legătură cu locul de stocare a datelor, utilizatorul nu trebuie să cunoască decât numele serverului. Iar serverul se poate afla în camera alăturată sau la 4000 de kilometri distanță. Acest aspect va fi rezolvat de rețea, nu de utilizator sau de baza de date.

9. Programele pentru aplicații și activitățile de la terminal rămân separate logic când sunt efectuate modificări de orice fel legate de păstrarea informațiilor, care permit separarea, asupra tabelelor de bază.

Indiciu

Programelor pentru aplicații nu ar trebui să li se solicite să cunoască modificările efectuate asupra structurilor de date. De exemplu, dacă sunt adăugate noi coloane în tabel, programul cu care lucrează acel tabel ar trebui să poată lucra neafectat.

---

10. Restricțiile de integritate specifice unei anumite baze de date relaționale trebuie să poată fi definite într-un sublimbaj relațional și stocabile în catalog, nu în programul pentru aplicații.

Indiciu

Menținerea integrității datelor este probabil una dintre cele mai importante funcții într-o bază de date relațională. Ca să fie relațional, un sistem de baze de date trebuie să poată să ofere aceste servicii. De aceea, o bază de date nu se poate baza pe programe disparate pentru a menține integritatea datelor. Dacă un câmp poate accepta numai anumite valori sau dacă datele dintr-un tabel nu pot fi șterse în cazul în care există alte date, acest lucru trebuie să poată fi definit într-o bază de date. Partea a treia a cărții tratează menținerea integrității datelor într-o bază de date.

---

11. Un DBMS relațional are independență distribuită (***distribution indepedence***).

Indiciu

O bază de date care respectă modelul relațional poate fi divizată sau partiționată de-a lungul mai multor locații fizice. În concluzie, o bază de date poate fi găzduită pe mai multe computere într-un mediu cu mai multe servere. Indiferent dacă o bază de date se află pe unul sau mai multe servere, programul sau utilizatorul final nu ar trebui să observe diferența.

---

12. Dacă un sistem relațional are un limbaj de nivel scăzut, acesta nu poate fi utilizat pentru a submina sau trece peste regulile de integritate și restricțiile exprimate în limbajul relațional de nivel înalt.

Indiciu

Limbajele de nivel scăzut au tendința să lucreze la o singură înregistrare o dată, pe când limbajele de nivel înalt sunt bazate pe seturi (pot lucra cu mai multe înregistrări o dată). Regula 12 stabilește de fapt că limbajele de nivel scăzut și cele de nivel înalt trebuie să respecte aceleași reguli.

În mod ironic, nici unul dintre sistemele de baze de date existente nu respectă toate cele 12 reguli. De aceea, prin definiție, un sistem de baze de date care este numit relațional nu este de fapt relațional. S-ar putea să spuneți: „Și ce dacă? Acesta este demonstrație a faptului că regulile sunt făcute pentru a fi încălcate." Aceasta este o observație bună. Înțelegerea posibilităților și a modului de lucru cu o bază de date relațională sunt mult mai importante

decât înțelegerea complexităților teoriei relaționale. Deși scopul acestei cărți îl reprezintă aspectele practice ale bazelor de date, înțelegerea originilor modelului bazei de date relaționale poate fi utilă. De aceea v-am oferit prezentarea celor 12 reguli ale lui Codd. În anexa E puteți găsi trimiteri la cărțile în care există informații avansate și puteți învăța mai multe despre teoria relațională.

Cariere în domeniul bazelor de date

Poate vă întrebați: „Cu aceste cunoștințe despre bazele de date pe care le voi obține, ce fel de slujbă voi obține?". Deoarece ați ales această carte, este probabil să fie adevărate două lucruri:

- Faptul că studiați bazele de date are legătură cu cariera dumneavoastră.
- Faptul că studiați bazele de date este explicat de motive personale.

Indiferent dacă interesul dumneavoastră pentru bazele de date este de natură personală sau profesională, această carte vă va oferi un punct de pornire pe calea cea bună. Poate că încercați să vă calificați într-o nouă meserie. Dar chiar dacă nu căutați un alt serviciu, după citirea cărții, s-ar putea să căpătați o mai mare curiozitate pentru un serviciu în acest domeniu. Iată o listă cu scurte descrieri ale posturilor din domeniul bazelor de date.

Administrator de baze de date

Un administrator de baze de date (***database administrator*** – DBA) este de obicei cel care administrează intrarea datelor într-o firmă. În mod normal, acest lucru necesită ca modificările structurilor de date, stocarea unor noi proceduri ș.a.m.d. să fie controlate de administrator. Acesta se asigură că bazele da date ale firmei sunt menținute corespunzător și funcționează optim. În timp, pe măsură ce o bază de date crește și se maturizează, implementarea curentă a acesteia trebuie reanalizată. Administratorul bazei de date joacă un rol central în acest proces.

Când este dezvoltată o nouă aplicație pentru o firmă, dezvoltatorul aplicației se consultă cu administratorul pentru a afla detaliile tehnice ale bazei de date. Deși dezvoltatorul bazei de date ar putea dori să-i aducă o modificare, probabil că și alte aplicații folosesc aceeași bază de date. Administratorul trebuie să se asigure că modificarea solicitată de o anumită aplicație este compatibilă cu alte aplicații. Dacă nu este compatibilă, administratorul trebuie să propună soluții alternative.

Programator de aplicații pentru baze de date

Poate că sunteți un programator cu experiență, dar nu ați mai lucrat cu date până acum. Cu abilitățile suplimentare pentru baze de date în arsenal, veți fi gata să atacați lumea în dezvoltare a aplicațiilor pentru baze de date. Spre deosebire de celelalte tipuri de aplicații software, aplicațiile pentru baze de date au un impact direct asupra operațiunilor unei firme.

Ca să dezvoltați o aplicație pentru baze de date destinată unei firme, trebuie mai întâi să înțelegeți activitatea acesteia. Acest lucru este total diferit de dezvoltarea unui utilitar care tratează un anumit aspect al activității firmei. Aplicațiile pentru baze de date tind să se ocupe de întreaga activitate a unei firme sau de un întreg segment al acestei activități. Dacă aveți experiență în dezvoltarea aplicațiilor bazate pe utilitare, aplicațiile pentru baze de date vor aduce la început cîteva noi provocări. Aceste noi provocări există deoarece cerințele sunt altele. O întrebare pe care este posibil să o puneți în acest moment este „În ce mod diferă cerințele?". Gândiți-vă la următorul lucru: programele bazate pe utilitare care îndeplinesc un singur scop au tendința de a fi limitate. Modul în care ele fac astăzi ceva va fi același și mâine. Bazele de date, pe de altă parte, cresc o dată cu firma. Mai mult, pe măsură ce mediul de afaceri se schimbă, cerințele unei aplicații pentru baze de date se schimbă și ele. Necesitatea de a regăsi noi informații, solicitarea unor noi rapoarte etc. (într-o măsură mai mică sau mai mare) necesită modificări asupra structurii bazei de date. S-a mai spus că o aplicație pentru baze de date nu este niciodată terminată, deoarece o firmă se schimbă în continuu.

Analist pentru baze de date

Poate că nu doriți să dezvoltați programe, ci mai degrabă sunteți o persoană care înțelege bine firma și datele legate de aceasta. Un post cheie în acest amalgam este cel de analist al bazei de date. Analiștii sunt planificatori cheie în designul unei baze de date. Este important de observat că dezvoltarea aplicațiilor și proiectarea lor sunt lucruri diferite. Primul și cel mai important pas în dezvoltarea unei aplicații pentru baze de date este proiectarea ei. Nu se poate face nimic altceva înainte de acest pas.

Rezumat

În acest capitol au fost prezentate elementele de bază despre bazele de date. Să recapituăm: o bază de date este o colecție de entități înrudite. Aceste entități, numite și tabele, sunt alcătuite din câmpuri, care mai sunt numite și atribute. Un tabel poate conține zero sau mai multe rânduri de date.

Ceea ce este o bază de date din punct de vedere tehnic nu este nici pe departe atât de interesant ca lucrurile pe care le poate face aceasta. Datele sunt sângele dătător de viață al unei firme. Toate lucrurile pe care o firmă dorește să le știe pot fi înmagazinate în date. Pentru a înțelege activitatea unei firme trebuie mai întâi să înțelegeți datele despre ea. Indiferent dacă datele sunt suportul unei campanii de marketing sau al serviciilor pentru clienți, datele joacă un rol important.

Azi, cele mai populare baze de date utilizate în firme au la bază modelul relațional dezvoltat de dr. E. F. Codd. Chiar dacă denumirea ***sistem de administrare a bazelor de date relaționale (relational database management system)*** este cuprinzătoare, nici un astfel de sistem aflat azi pe piață nu respectă toate cele 12 reguli ale lui Codd. Distincția poate fi de natură semantică,

dar este totuși utilă pentru a înțelege fundamentul pe care sunt construite bazele de date relaționale.

Întrucât bazele de date sunt sângele dătător de viață al unei firme, există multe cariere interesante asociate bazelor de date. Trei dintre acestea – administrator al bazei de date, progamator de aplicații pentru baze de date și analist al bazei de date – au fost decrise pe scurt în acest capitol.

Acum, că știți în mare ce este o bază de date, sunteți gata pentru pasul următor: vă veți „murdări" mâinile proiectând și construind prima bază de date.



2

ANATOMIA UNEI BAZE DE DATE

În acest capitol:

- O introducere în Microsoft Access
- Baza de date Northwind Traders
- Rezumat

Capitolul 1 a fost dedicat aspectelor teoretice despre bazele de date. Acest capitol și celelalte ale acestei părți au ca subiect aspectele practice ale bazelor de date. Bazele de date sunt entități tangibile, pe care le puteți vedea și cu care puteți lucra. Deși teoria este importantă și utilă, nu există metodă mai bună de învățare decât examinarea unei baze de date reale, inspectarea conținutului ei și înțelegerea modului în care a fost construită.

Acest capitol examinează baza de date Northwind Traders, care este livrată împreună cu Microsoft Access. Pentru a exploata la maxim acest capitol și restul cărții, ar trebui să aveți programul Access instalat pe computerul dumneavoastră. Vă rugăm să revedeți cerințele privitoare la software din secțiunea introductivă pentru detalii despre modul în care puteți obține și instala Access.

Acest capitol nu își propune să fie o trecere în revistă cuprinzătoare a întregului program Access. Mai curând, scopul capitolului este de a fi un ghid rapid despre funcționarea programului Access și de a vă învăța să fiți productiv în acest mediu de lucru. Multe dintre sursele de studiu disponibile azi vă pot învăța toate amănuntele despre Microsoft Access.

Înainte de a merge mai departe, ar trebui să vă asigurați că ați înțeles bine cele discutate în acest capitol.

Notă

Câteva dintre resursele populare de unde puteți afla informații despre Microsoft Access sunt:

- Site-ul oficial al Microsoft Access, `http://www.microsoft.com/office.access`
- Grupul de discuții oficial pentru suport Access la adresa `http:/communities.microsoft.com/newsgroups/default.asp?icp=gss&slcid=us`
- Site-ul oficial Microsoft Knowledge Base, la `http://www.microsoft.com/KB/`

Introducere în Microsoft Access

Access, care face parte din suita Microsoft Office, este un produs complet pentru baze de date. În afară de faptul că este complet, este foarte ușor de utilizat. Primul pas în acest proces este lansarea lui. Figura 2.1 ilustrează opțiunile din meniu necesare pentru lansare.

După lansarea programului Access, veți vedea o casetă de dialog care vă invită să creați o nouă bază de date, să lansați un wizard din Access, sau să deschideți o bază de date existentă (vedeți figura 2.2).



FIGURA 2.1
Puteți lansa Access selectând Start, Programs, Microsoft Access din meniul principal.

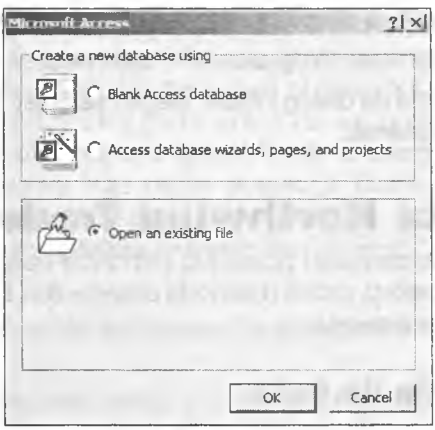


FIGURA 2.2
După lansarea programului Access, apare caseta de dialog inițială, care vă permite să creați sau să deschideți o bază de date.

În acest caz, veți deschide un fișier existent executând clic pe opțiunea corespunzătoare, apoi pe OK. Apoi, deschideți fișierul `\Program Files\Microsoft Access\Office\Samples\Northwind.mdb`. Extensia implicită pentru un fișier Access este MDB.

După ce ai deschis baza de date Northwind Traders, este afișat formularul principal al bazei de date – care oferă acces la toate componentele bazei de date – vedeți figura 2.3.

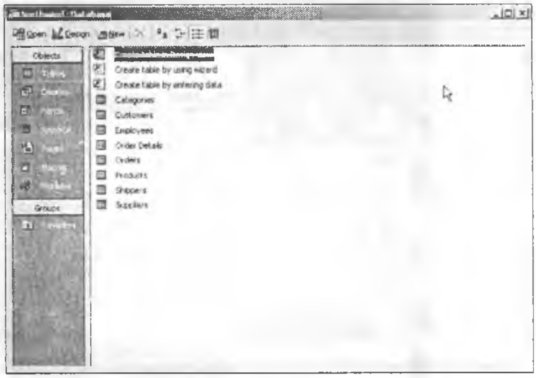


FIGURA 2.3
Caseta de dialog principală a bazei de date oferă acces la toate componentele bazei de date, inclusiv tabele, interogări, formulare și rapoarte.

> **Notă** Access vă oferă și posbilitatea de a scrie pagini Web, macrocomenzi și module de cod. Deși veți vedea referințe la acestea în diferite ilustrații, ele depășesc scopul acestei cărți. Pentru mai multe informații despre ele, citiți nota de la începutul capitolului privind sursele populare de informare pentru Access.

Iată o recapitulare despre lansarea Microsoft Access:

1. Selectați Start, Programs, Microsoft Access.
2. Din caseta de dialog inițială, alegeți să creați sau selectați o bază de date existentă.

Baza de date Northwind Traders

Figura 2.3 prezintă formularul principal pentru o bază de date din Access. Din acest formular, puteți vedea diferitele obiecte din baza de date, precum și datele conținute de aceasta.

Relațiile din baza de date

Pentru a înțelege scopurile unei baze de date, va trebui să vedeți relațiile dintre diferitele entități ale acesteia. Figura 2.4 prezintă vizualizarea conținând relațiile dintr-o bază de date. Pentru a avea acces la această vizualizare, va trebui să executați clic pe butonul Relationship View (prezentat și el în figura 2.4).

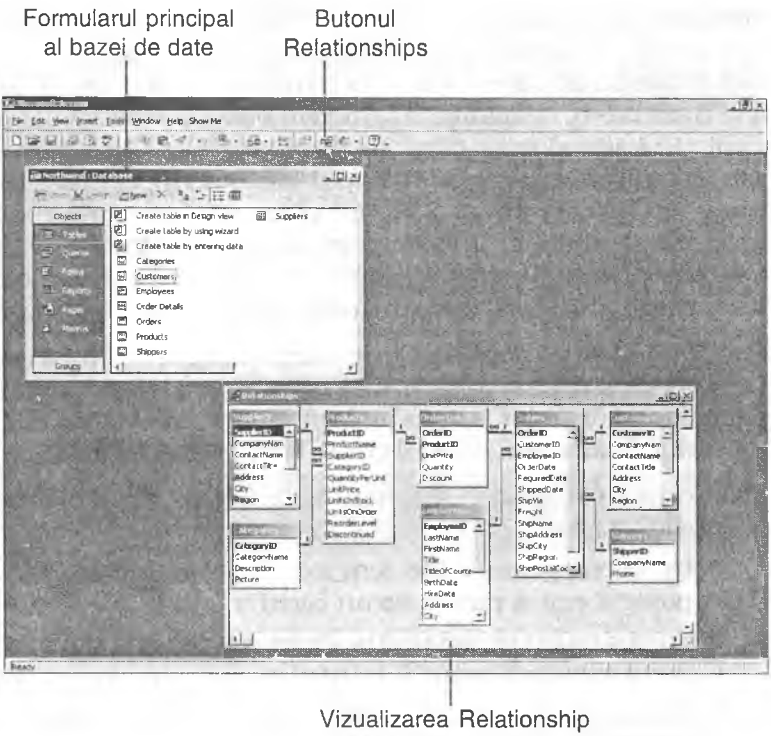


FIGURA 2.4
Vizualizarea relațiilor vă oferă posibilitatea de a vedea toate tabelele dintr-o bază de date, precum și modul în care sunt acestea legate.

Dacă studiați diagrama bazei de date ilustrată în figura 2.4, scopul bazei de date Northwind Traders devine clar. Dacă priviți din partea dreaptă a diagramei, tabelul Customers (clienți) este conectat cu tabelul Orders (comenzi). Lângă tabelul Customers este numărul 1, iar lângă tabelul Orders este simbolul infinitului, care arată ca un 8 așezat pe orizontală. O astfel de relație este numită ***unu la mulți (one-to-many)***. Cu alte cuvinte, un client poate da zero sau mai multe comenzi. Dacă priviți relația în sens invers, o comandă poate fi dată doar de un client. În acst caz, tabelul Customers este considerat ***părinte (parent)***, iar tabelul Orders este considerat ***copil (child)***.

Cele trei tipuri de relații pe care le veți întâlni când lucrați cu bazele de date sunt următoarele:

- ***Unu la unu (one-to-one)*** – O înregistrare din tabelul părinte este înrudită numai cu o înregistare din tabelul copil.
- ***Unu la mulți (one-to-many)*** – O înregistrare din tabelul părinte este înrudită cu zero sau mai multe înregistrări din tabelul copil.
- ***Mulți la mulți (many-to-many)*** – Mai multe înregistrări din tabelul părinte sunt înrudite cu mai multe înregistrări din tabelul copil. Relațiile de acest tip sunt de fapt sparte în mai multe relații unu la mulți. Acest tip de relații este explicat în nota următoare.

Funcționarea relațiilor este foarte simplă. Dacă priviți toate tabelele din figura 2.4, veți observa că unele câmpuri sunt scrise cu litere îngroșate (bold). Combinația acestor tipuri alcătuiește cheia primară. Reluând conceptele discutate în capitolul 1, o ***cheie primară*** este identificatorul unic al unui rând sau al unui tabel. În cazul tabelului Customers, cheia primară sau identificatorul unic este câmpul CustomerID (identificatorul clientului). Dacă veți privi la figura 2.7, veți vedea că fiecare rând care corespunde unui client are o altă valoare în câmpul CustomerID. Aceeași valoare CustomerID este preluată în tabelul Orders. Iată cum o bază de date știe care înregistrare copil aparține unei anumite înregistrări părinte.

Continuând să vă deplasați spre stânga în figura 2.4, veți vedea că tabelul Orders este și copil și părinte. Este copil pentru tabelul Customers și părinte pentru tabelul Order Details (informații privind comanda). Din perspectiva tabelului Customers, tabelul Order Details este nepot.

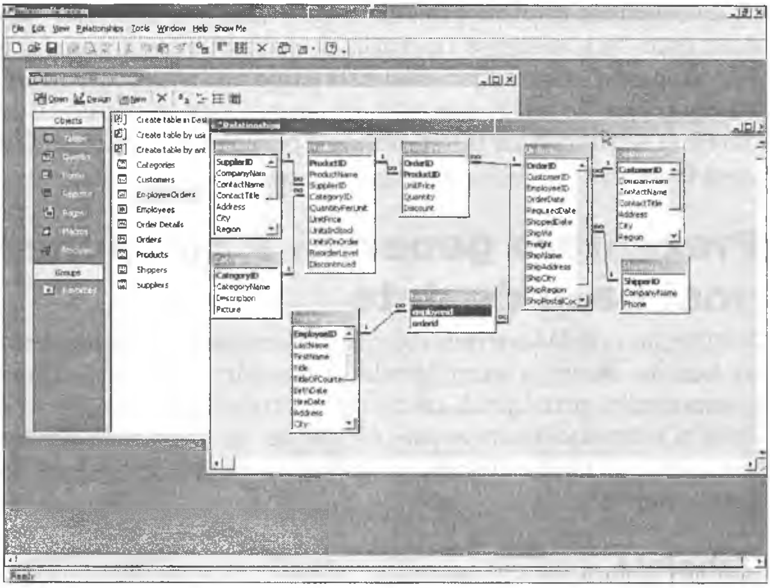
Propozițiile următoare sintetizează relațiile conținute în baza de date Northwind Traders:

- Un client poate emite zero sau mai multe comenzi, iar o comandă poate fi emisă numai de un client.
- Un cărăuș (transportator) poate transporta una sau mai multe comenzi, dar o comandă poate fi transportată de un singur cărăuș.
- Un angajat poate prelua zero sau mai multe comenzi, dar o comandă poate fi preluată numai de un angajat.

Să examinăm relația dintre tabelele Orders și Employee (angajați). Așa cum este proiectată baza de date, o comandă poate fi preluată numai de către un angajat. Cum ar fi dacă o comandă ar putea fi preluată de mai mulți angajați? Poate că firma folosește echipe de angajați pentru a prelua comenzile. În acest caz, designul curent al bazei de date nu respectă această cerință. Aici intervine conceptul de relație mulți la mulți.

Să presupunem că ar fi nevoie de mai mulți angajați pentru a prelua o comandă. Dacă examinăm relațiile curente, tabelul Employee este părinte, iar tabelul Orders este copil. Știți deja că există un câmp EmployeeID în tabelul Orders și în acest câmp poate fi stocată o valoare unică. Cum poate fi realizat acest lucru? Răspunsul constă în adăugarea unui tabel suplimentar numit EmployeeOrders. Figura 2.5 prezintă o versiune modificată a bazei de date Northwind Traders. Pentru a îndeplini această sarcină, deoarece modelul relațional nu poate accepta direct o relație mulți la mulți, această relație este indirect suportată folosind două relații unu la mulți. Tabelul intermediar este numit tabel de joncțiune (***junction table***).

FIGURA 2.5
Relațiile mulți la mulți sunt suportate prin crearea unui tabel intermediar, numit tabel de joncțiune.



Pentru a continua exemplificarea, să presupunem că în preluarea unei anumite comenzi sunt implicați doi angajați. În acest caz, vor apărea două înregistrări în tabelul de joncțiune. Din perspectiva comenzii, veți vedea doi angajați implicați în preluarea acesteia. Din perspectiva unui angajat, veți vedea toate comenzile în preluarea cărora a fost implicat angajatul respectiv. În acest scenariu, aceeași comandă poate fi asociată cu mai mulți angajați.

Pe scurt, următoarele fraze descriu esența relației mulți la mulți:

- O comandă poate conține una sau mai multe informații, dar o informație despre comezi poate aparține unei singure comenzi.
- Un produs poate apărea în zero sau mai multe informații despre comenzi, dar o informație despre comenzi poate conține un singur produs.
- Un furnizor poate livra zero sau mai multe produse, dar un produs poate fi livrat de un singur furnizor.
- Categoriile pot fi alcătuite din zero sau mai multe produse, dar un produs poate aparține unei singure categorii.

Notă

Dar dacă un produs ar putea aparține mai multor categorii? Dacă vă gândiți că o relație mulți la mulți poate fi mascată printr-un tabel de joncțiune, sunteți pe calea cea bună și înseamnă că învățați repede!

Dacă revedem relațiile din Northwind Traders, este clar că scopul bazei de date este de a suporta operația de introducere a comenzilor. În afară de introducerea comenzilor, mai este suportat și un sistem de administrare a inventarului de bază. Așa cum am menționat în capitolul 1, dacă înțelegeți datele și structura lor puteți obține o perspectivă imediată inițială a activității unei firme.

Prezentare generală a componentelor unei baze de date

Secțiunile următoare descriu componentele principale ale unei baze de date în Access. Acestea sunt tabele, interogări, rapoarte și formulare. Fiecare componentă principală are variate subcomponente, iar aici vom prezenta doar o introducere în aceste elemente. În capitolele următoare vor fi prezentate mai multe detalii, pe măsură ce veți începe să lucrați cu aceste componente.

Tabelele

Să presupunem, de exemplu, că doriți să vedeți conținutul tabelului Customers. Cum ați putea realiza acest lucru? E foarte simplu, dacă parcurgeți pașii următori (vedeți figura 2.6):

1. Selectați tabelul Customers cu un clic de mouse.
2. După selectare, executați clic pe butonul Open de pe bara de instrumente.

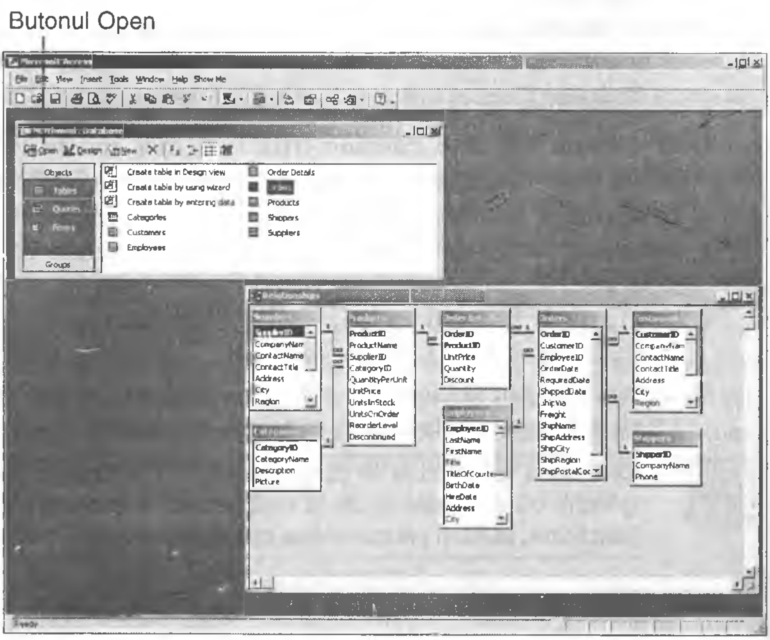


FIGURA 2.6
Puteți vedea datele conținute într-un tabel dacă deschideți tabelul.

Figura 2.7 prezintă conținutul tabelului Customers. Această vizualizare se numește Datasheet (foaie de date). Denumirea este firească, deoarece seamănă cu o foaie de calcul din Excel.

FIGURA 2.7

În vizualizarea Datasheet datele pot fi văzute, modificate sau șterse.

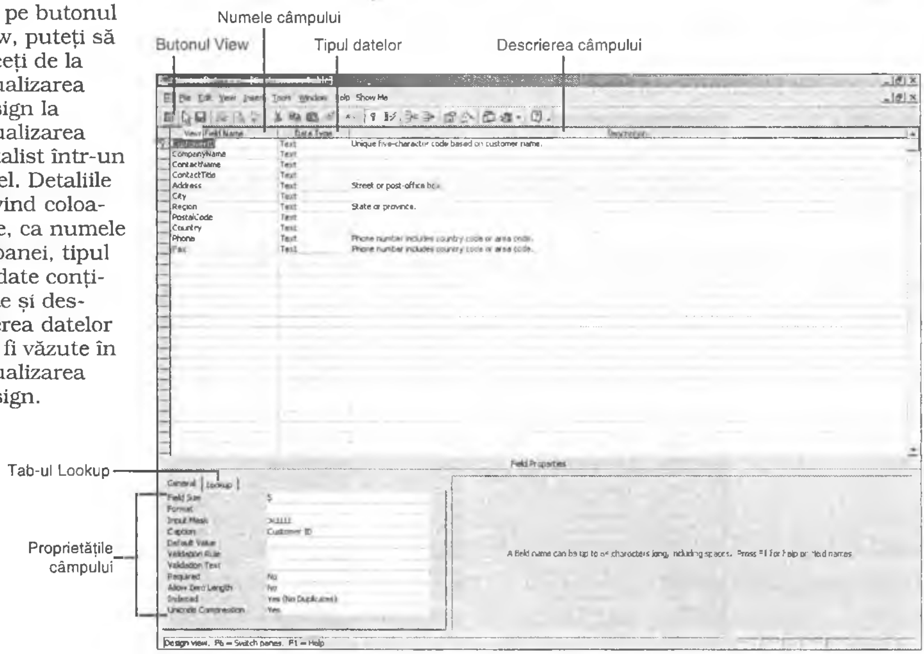


Atenție Când vedeți datele în vizualizarea Datasheet, este important să observați că aveți în față o redare directă (***live view***) a datelor. Aceasta înseamnă că datele pot fi modificate sau șterse. Nu uitați, fiți atent la tastele pe care le apăsați când vizualizați datele.

Care este structura unui tabel? Vă amintiți discuția din capitloul 1, dspre tabele? Tabe'ele sunt formate din una sau mai multe coloane. Cum poate fi vizualizată structura unui tabel în Access? Figura 2.7 și pașii următori vă pot explica acest lucru:

1. Găsți butonul View, ultimul din stânga pe bara de instrumente (vedeți figura 2.8).
2. Executați clic pe butonul View. Va fi afișată vizualizarea Design.
3. Executați din nou clic pe butonul View; va fi afișată vizualizarea Datalist (lista datelor). Aceasta reprezintă o altă modalitate de a proiecta un tabel, dar nu este la fel de utilă ca vizualizarea Design.

FIGURA 2.8
Dacă executați clic pe butonul View, puteți să treceți de la vizualizarea Design la vizualizarea Datalist într-un tabel. Detaliile privind coloanele, ca numele coloanei, tipul de date conținute și descrierea datelor pot fi văzute în vizualizarea Design.



Atenție

Este important să fiți foarte atent când vizualizați datele dintr-un tabel. Pe măsură ce derulați în sus și în jos lista câmpurilor, este foarte ușor să modificați din greșeală atributele unui câmp. Nu uitați – fiți atent ce taste apăsați când vizualizați structura tabelului!

Notă

Scopul și utilizarea tab-ului lookup sunt discutate mai detaliat în secțiunea referitoare la formulare din acest capitol. Tab-urile Lookup oferă o cale prietenoasă față de utilizator pentru asocierea datelor din tabele înrudite.

Proprietăți

În vizualizarea Design, dacă derulați în sus și în jos lista câmpurilor, puteți vedea proprietățile extinse ale fiecărui câmp. În afară de nume, tipul de date și descrierea acestora, alte atribute ale câmpurilor – ca, mărime, format, mască de intrare și legendă – pot fi vizualizate. Aceste atribute extinse pot fi vizualizate în partea din stânga jos a figurii 2.7.

FieldName (numele câmpului) Proprietatea FieldName precizează numele unui câmp al tabelului. Numele unui câmp poate avea lungimea maximă de 64 de caractere și poate conține orice combinație de litere, numere, spații și caractere speciale. Caracterele nepermise sunt punctul, semnul exclamării, accentul și parantezele. Credeți sau nu, ghilimelele duble sunt permise în numele unui câmp, dar este îndoielnic că această posibilitate ar putea fi considerată utilă de cineva. De regulă, veți folosi literele alfabetului, cifrele de la 0 la 9 și caracterul underscore (linie jos).

Atenție

Deși poate că ați inclus spații în numele câmpului, ele vă pot face mai multe probleme decât merită. De exemplu, numele firmei Northwind Traders din tabelul Northwind Customers. Dacă utilizați spații, va trebui să referiți câmpul sub forma `[customers.numele firmei]`. Altfel, dacă nu folosiți spații, câmpul poate fi referit astfel: `customers.numelefirmei`. Folosirea spațiilor tinde să complice lururile în loc să le simplifice, deci evitați spațiile în numele câmpurilor.

DataType (tipul datelor) Proprietatea `DataType` specifică tipul de date conținute în câmpul respectiv. Tipurile standard de date sunt prezentate în tabelul 2.1

Tabelul 2.1 Tipurile standard de date suportate de Access

Tipul de date	Descriere	Dimensiunea maximă
Text	Date de tip caracter	255 bytes
Memo	Date de tip caracter	64 K
Number	Date numerice	8 bytes
Date/Time	Date de tip dată calendaristică și oră	8 bytes
Currency	Date monetare	8 bytes
Autonumber	Date numerice cu autoincrementare	4 bytes
Yes/No	Date logice de tip True/False (adevărat/fals)	1 bit

Notă

Access suportă și alte tipuri de date, inclusiv OLE Object și Hyperlink. Pentru mai multe informații despre aceste tipuri de date, consultați sursele de documentare despre Access.

Desription (descriere) Proprietatea Description vă permite să oferiți informații descriptive despre câmpul respectiv. O descriere poate avea cel mult 255 de caractere.

Field Size (dimensiunea câmpului) Proprietatea Field Size vă permite să specificați deimensiunea maximă a datelor stocate în acel câmp. Vedeți tabelul 2.1 pentru valorile maxime ale dimensiunilor câmpurilor.

Format (format) Proprietatea Format controlează modul de afișare a datelor. Această proprietate vă permite să stabiliți modul în care apar datele numerice, monetare, de tip text și cele de tip dată calendaristică sau oră.

Input Mask (mască de intrare) Proprietatea Input Mask oferă un mecanism pentru introducerea mai ușoară a datelor. De exemplu, pentru un număr de telefon, o mască de tipul (____)- (______) ușurează introducerea diferitelor componente ale numărului de telefon. Access oferă un program wizard pentru a crea valorile pentru această proprietate.

Caption (legenda) Proprietatea Caption determină conținutul descriptiv al unei etichete folosite într-un formular sau raport care conține câmpul respectiv. Proprietatea Caption precizează și conținutul titlului unei coloane pentru un câmp în vizualizarea Datalist. (Vedeți figura 2.7 pentru o prezentare a vizualizării Datalist.)

Default Value (valoare implicită) Proprietatea Default Value specifică valoarea introdusă automat în câmp când este creată o nouă înregistrare. De exemplu, un câmp cu date numerice poate avea valoarea implicită 0, iar un câmp cu date calendaristice poate avea data curentă ca valoare implicită.

Validation Rule (regulă de validare) Proprietatea Validation Rule specifică o condiție pe care trebuie să o îndeplinească valoarea introdusă. De exemplu, o valoare numerică trebuie să fie >=0. Regulile de validare ajută la păstrarea integrității datelor.

Validation Text (text de validare) Proprietatea Validation Text specifică mesajul sub formă de text afișat pentru utilizatorul care nu respectă regula de validare. Dimensiunea maximă a proprietății este de 255 de caractere.

Rsquired (necesar) Proprietatea Required specifică dacă valoarea unei date este necesară într-un câmp.

Allow Zero Length (acceptă lungimea zero) Proprietatea Allow Zero Length specifică dacă este permisă introducerea unui șir de caractere cu lungime zero. Această proprietate poate fi aplicată câmpurilor cu date de tip text, memo și hyperlink.

Indexed (indexat) Proprietatea Indexed specifică dacă este creat un index pentru câmpul respectiv. Pot fi create două tipuri de index. Unul permite introduceri duplicate, iar celălalt nu acceptă duplicatele.

Compresia Unicode

Access suportă reprezentarea Unicode a caracterelor, ceea ce înseamnă că acestea sunt reprezentate prin doi biți în loc de unul. Datorită reprezentării

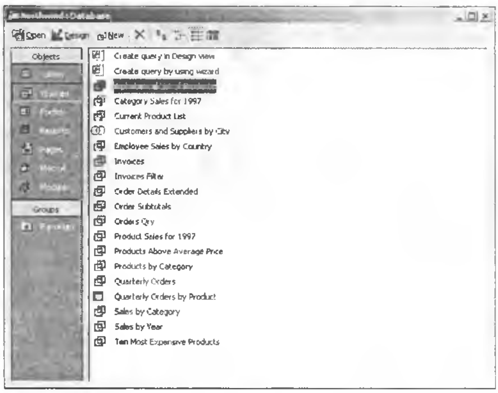
pe doi biți pot fi suportate mai multe pagini de cod a limbajului. Pe de altă parte, în reprezentarea pe un bit este suportată o singură pagină de cod și există cerințe mai mici privind stocarea pe disc.

Interogări

Interogările oferă o cale de vizualizare a datelor. De cele mai multe ori, va trebui să utilizați și să interacționați cu datele în altă manieră decât cea în care le stocați. Figura 2.9 prezintă interogările definite în baza de date Northwind Traders.

FIGURA 2.9

O mare varietate de interogări sunt definite în baza de date Northwind Traders.



Cea mai bună cale de a prezenta interogările este un exemplu simplu. Cerința este o listă alfabetică a produselor active. Pentru o astfel de interogare, aveți nevoie numai de două informații: identificatorul produsului (product ID) și numele produsului. Interogarea Current Product List (lista produselor curente) îndeplinește aceste cerințe. Pașii necesari pentru vizualizarea coținutului unei interogări sunt exact aceiași ca pentru vizualizarea conținutului unui tabel:

1. Selectați interogarea Current Product List executând clic pe ea.
2. După selectare, executați clic pe butonul Open din bara de instrumente.

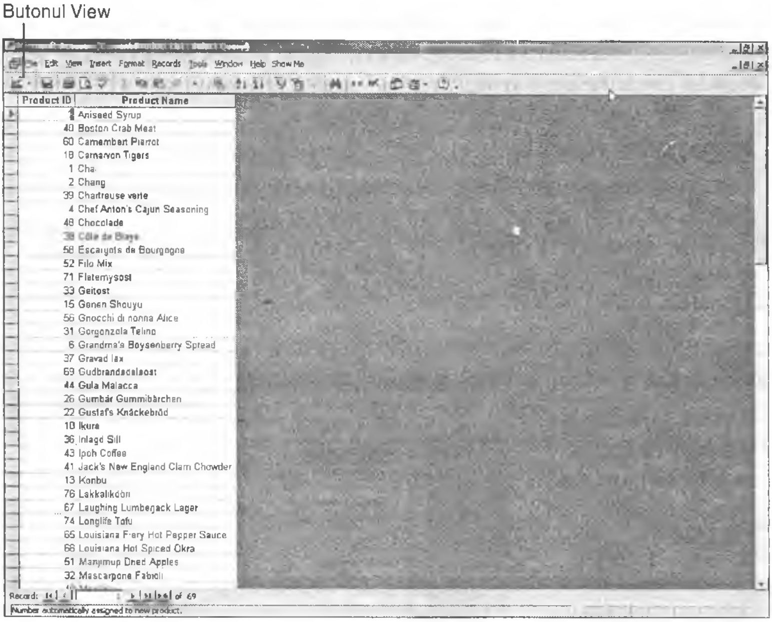
Figura 2.10 prezintă rezultatele lansării interogării Current Product List.

Una dintre întrebările dumneavoastră ar putea fi: "Care este structura unei interogări?". Structura unei interogări este derivată din structura tabelelor care furnizează informațiile pentru interogare. Pașii necesari pentru vizualizarea structurii unei interogări sunt aceiași ca pentru un tabel:

1. Executați clic pe butonul View. Veți trece astfel în vizualizarea Design.
2. Executați din nou clic pe butonul View; veți trece astfel în vizualizarea Datalist.

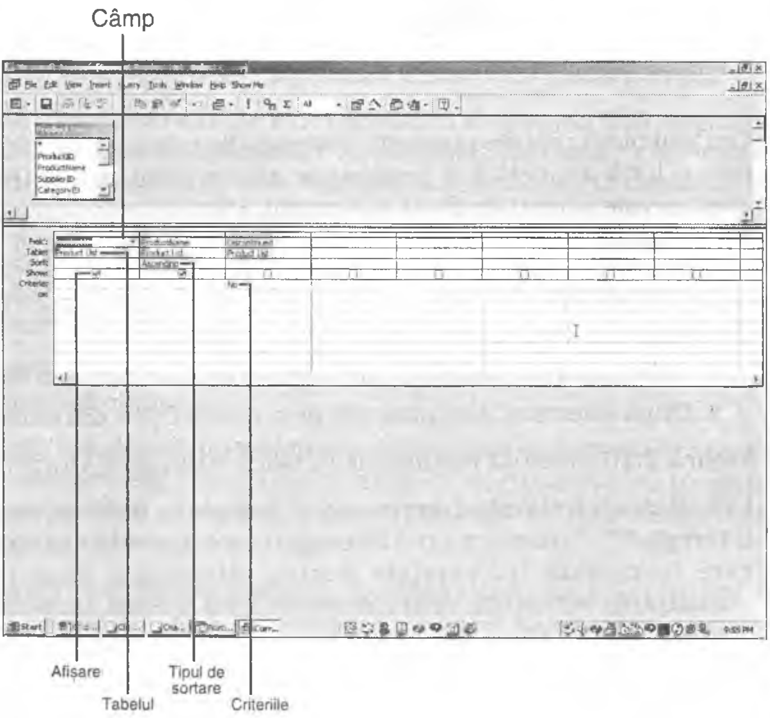
Structura interogării Current Product List este prezentată în figura 2.11.

FIGURA 2.10



Interogarea Current Product List oferă o listă alfabetică a produselor active.

FIGURA 2.11



Structura unei interogări specifică tabelele și coloanele utilizate, ordinea de sortare și criteriul de selecție.

În interogarea Current Product List sunt utilizate trei câmpuri, dar numai două sunt afișate în setul de rezultate. Al treilea câmp, Discontinued (discontinuu), este utilizat numai în scopul stabilirii criteriilor. În acest caz, sunt selectate numai înregistrările a căror valoare Discontinued este no.

Unul dintre cele două câmpuri afișate, Product Name (numele produsului) este și el afișat și folosit ca bază de sortare. În acest caz, setul de rezultate este ordonat după numele produsului în ordine ascendentă (crescătoare).

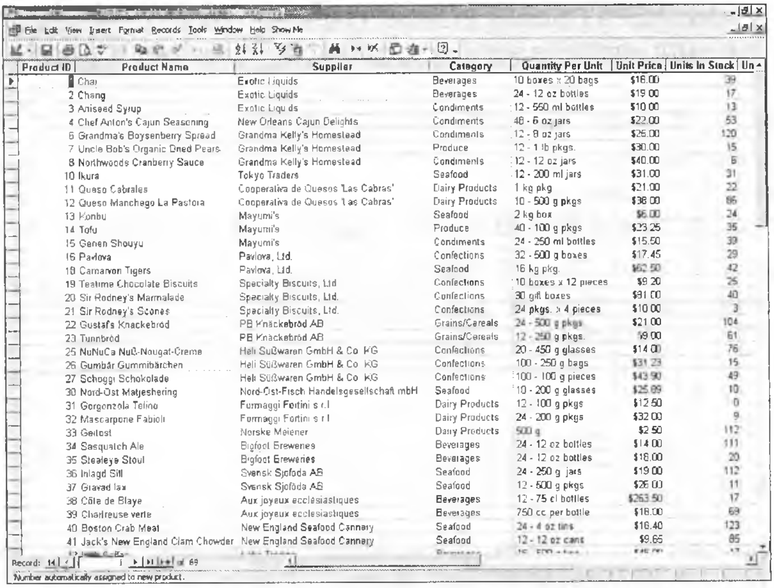
Interogările care conțin date din mai multe tabele nu sunt substanțial diferite de cele care conțin date dintr-un singur tabel. Un exemplu simplu în acest sens este interogarea Alphabetical List of Products (lista alfabetică a produselor). În această interogare, datele din tabelele produselor și categoriilor sunt combinate pentru a forma un singur set de rezultate (vedeți figura 2.12).

FIGURA 2.12

Tabelele din interogările multitabel sunt automat înrudite, pe baza relațiilor definite în baza de date.



Setul de rezultate al interogării Alphabetical List of Products este prezentat în figura 2.13.



Product ID	Product Name	Supplier	Category	Quantity Per Unit	Unit Price	Units In Stock
1	Chai	Exotic Liquids	Beverages	10 boxes x 20 bags	$18.00	39
2	Chang	Exotic Liquids	Beverages	24 - 12 oz bottles	$19.00	17
3	Aniseed Syrup	Exotic Liquids	Condiments	12 - 550 ml bottles	$10.00	13
4	Chef Anton's Cajun Seasoning	New Orleans Cajun Delights	Condiments	48 - 6 oz jars	$22.00	53
6	Grandma's Boysenberry Spread	Grandma Kelly's Homestead	Condiments	12 - 8 oz jars	$25.00	120
7	Uncle Bob's Organic Dried Pears	Grandma Kelly's Homestead	Produce	12 - 1 lb pkgs.	$30.00	15
8	Northwoods Cranberry Sauce	Grandma Kelly's Homestead	Condiments	12 - 12 oz jars	$40.00	6
10	Ikura	Tokyo Traders	Seafood	12 - 200 ml jars	$31.00	31
11	Queso Cabrales	Cooperativa de Quesos 'Las Cabras'	Dairy Products	1 kg pkg	$21.00	22
12	Queso Manchego La Pastora	Cooperativa de Quesos 'Las Cabras'	Dairy Products	10 - 500 g pkgs	$38.00	86
13	Konbu	Mayumi's	Seafood	2 kg box	$6.00	24
14	Tofu	Mayumi's	Produce	40 - 100 g pkgs	$23.25	35
15	Genen Shouyu	Mayumi's	Condiments	24 - 250 ml bottles	$15.50	39
16	Pavlova	Pavlova, Ltd.	Confections	32 - 500 g boxes	$17.45	29
18	Carnarvon Tigers	Pavlova, Ltd.	Seafood	16 kg pkg.	$62.50	42
19	Teatime Chocolate Biscuits	Specialty Biscuits, Ltd.	Confections	10 boxes x 12 pieces	$9.20	25
20	Sir Rodney's Marmalade	Specialty Biscuits, Ltd.	Confections	30 gift boxes	$81.00	40
21	Sir Rodney's Scones	Specialty Biscuits, Ltd.	Confections	24 pkgs. x 4 pieces	$10.00	3
22	Gustaf's Knäckebröd	PB Knäckebröd AB	Grains/Cereals	24 - 500 g pkgs	$21.00	104
23	Tunnbröd	PB Knäckebröd AB	Grains/Cereals	12 - 250 g pkgs.	$9.00	61
25	NuNuCa Nuß-Nougat-Creme	Heli Süßwaren GmbH & Co. KG	Confections	20 - 450 g glasses	$14.00	76
26	Gumbär Gummibärchen	Heli Süßwaren GmbH & Co. KG	Confections	100 - 250 g bags	$31.23	15
27	Schoggi Schokolade	Heli Süßwaren GmbH & Co. KG	Confections	100 - 100 g pieces	$43.90	49
30	Nord-Ost Matjeshering	Nord-Ost-Fisch Handelsgesellschaft mbH	Seafood	10 - 200 g glasses	$25.89	10
31	Gorgonzola Telino	Formaggi Fortini s.r.l.	Dairy Products	12 - 100 g pkgs	$12.50	0
32	Mascarpone Fabioli	Formaggi Fortini s.r.l.	Dairy Products	24 - 200 g pkgs	$32.00	9
33	Geitost	Norske Meierier	Dairy Products	500 g	$2.50	112
34	Sasquatch Ale	Bigfoot Breweries	Beverages	24 - 12 oz bottles	$14.00	111
35	Steeleye Stout	Bigfoot Breweries	Beverages	24 - 12 oz bottles	$18.00	20
36	Inlagd Sill	Svensk Sjöföda AB	Seafood	24 - 250 g jars	$19.00	112
37	Gravad lax	Svensk Sjöföda AB	Seafood	12 - 500 g pkgs	$26.00	11
38	Côte de Blaye	Aux joyeux ecclésiastiques	Beverages	12 - 75 cl bottles	$263.50	17
39	Chartreuse verte	Aux joyeux ecclésiastiques	Beverages	750 cc per bottle	$18.00	69
40	Boston Crab Meat	New England Seafood Cannery	Seafood	24 - 4 oz tins	$18.40	123
41	Jack's New England Clam Chowder	New England Seafood Cannery	Seafood	12 - 12 oz cans	$9.65	85

FIGURA 2.13
Rezultatul unei interogări multitabel (cu date din mai multe tabele) apar sub forma unui tabel.

Rapoarte

Informațiile dintr-o bază de date nu sunt prea utile dacă nu pot fi accesate și folosite într-o manieră prietenoasă față de utilizator. Când vă îmbarcați în procesul dezvoltării aplicațiilor pentru baze de date, în mod invariabil veți fi fi întrebat ce rapoarte pot fi generate. În baza de date Northwind Traders sunt definite mai multe rapoarte (vedeți figura 2.14).

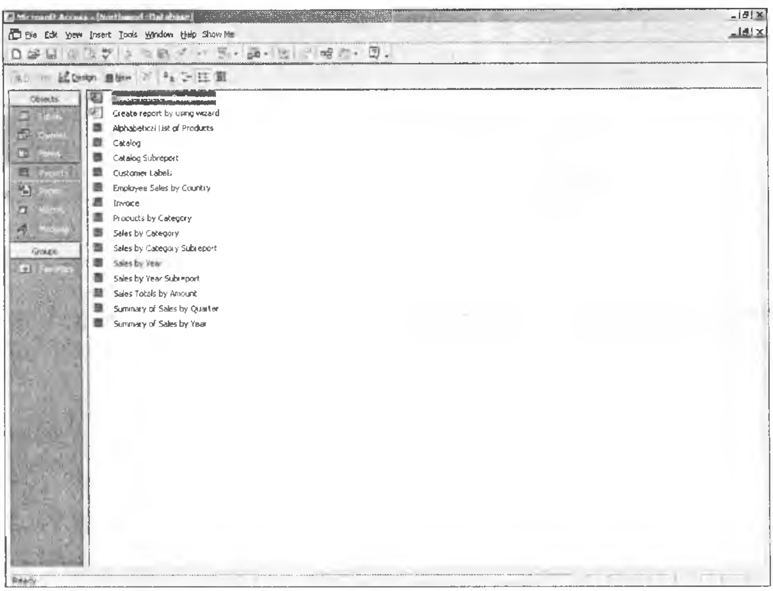


FIGURA 2.14
Rapoartele definite în baza de date Northwind Traders.

Pentru a explica funcționarea rapoartelor, să considerăm interogarea Products by Category (produse clasificate pe categorii). Această interogare este similară celor examinate în secțiunea anterioară și este prezentată în figura 2.15.

Interogarea Products by Category afișează informațiile prezentate în tabelul 2.2.

Tabelul 2.2 Structura interogării Products by Category

Numele câmpului	Tabelul	Sortare/criterii
Category Name	Categories	Ascending
Product Name	Products	Ascending
Quantity per Unit	Products	
Units in Stock	Products	
Discontinued	Products	<>Yes

Rezultatul interogării Products by Category este prezentat în figura 2.16.

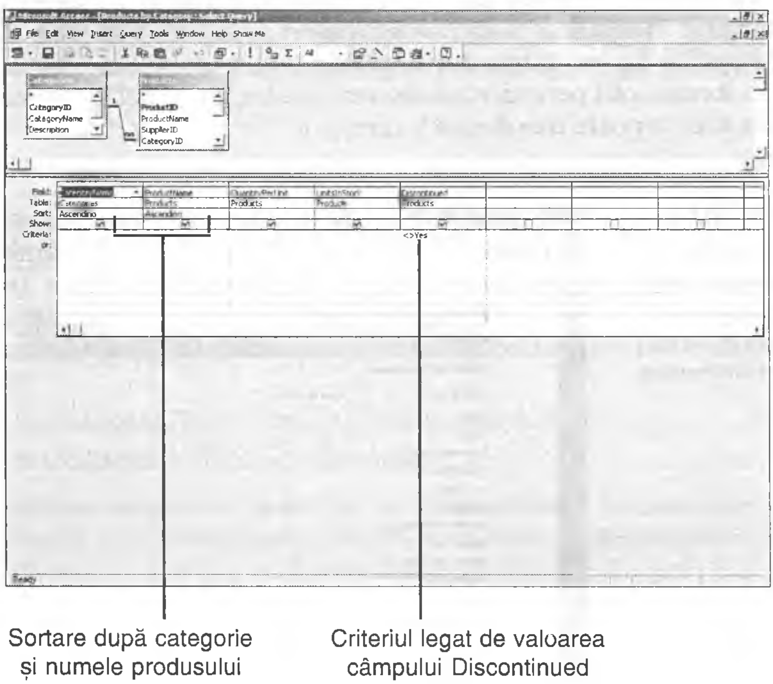
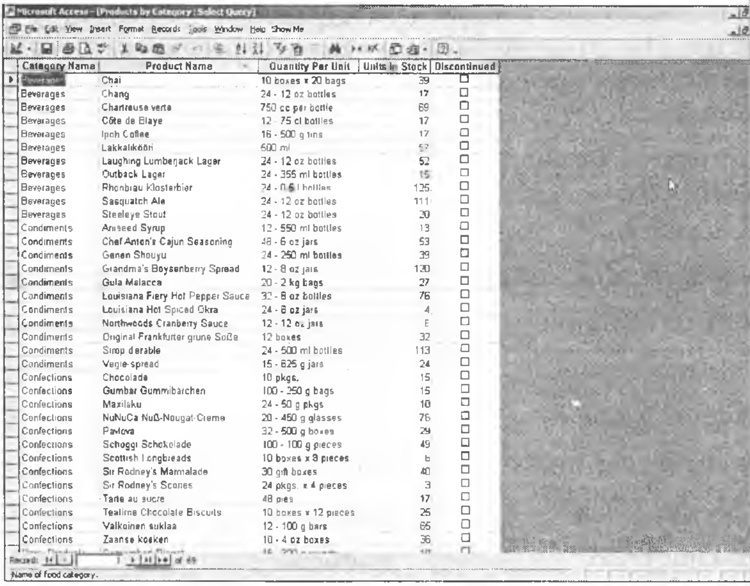


FIGURA 2.15
Interogarea Products by Category în vizualizarea Design.

FIGURA 2.16

Interogarea Products by Category în vizualizarea Datalist.

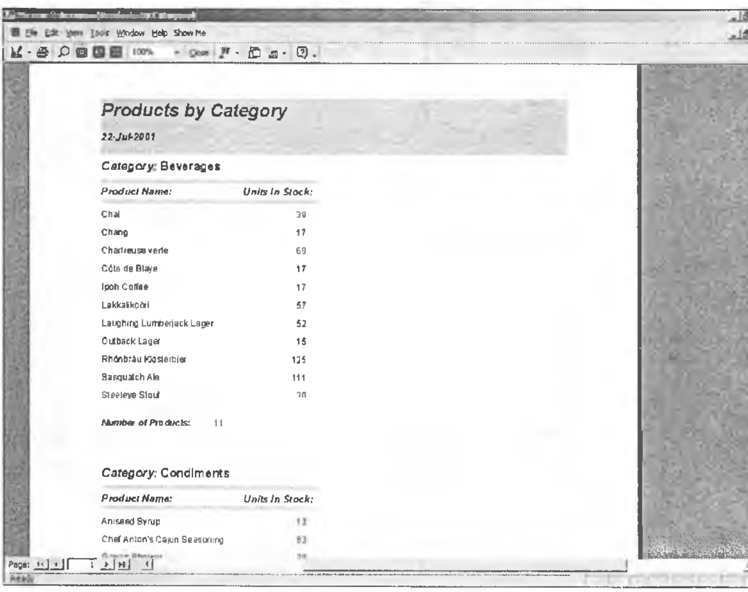


Category Name	Product Name	Quantity Per Unit	Units In Stock	Discontinued
Beverages	Chai	10 boxes x 20 bags	39	☐
Beverages	Chang	24 - 12 oz bottles	17	☐
Beverages	Chartreuse verte	750 cc per bottle	69	☐
Beverages	Côte de Blaye	12 - 75 cl bottles	17	☐
Beverages	Ipoh Coffee	16 - 500 g tins	17	☐
Beverages	Lakkalikööri	500 ml	57	☐
Beverages	Laughing Lumberjack Lager	24 - 12 oz bottles	52	☐
Beverages	Outback Lager	24 - 355 ml bottles	15	☐
Beverages	Rhönbräu Klosterbier	24 - 0.5 l bottles	125	☐
Beverages	Sasquatch Ale	24 - 12 oz bottles	111	☐
Beverages	Steeleye Stout	24 - 12 oz bottles	20	☐
Condiments	Aniseed Syrup	12 - 550 ml bottles	13	☐
Condiments	Chef Anton's Cajun Seasoning	48 - 6 oz jars	53	☐
Condiments	Genen Shouyu	24 - 250 ml bottles	39	☐
Condiments	Grandma's Boysenberry Spread	12 - 8 oz jars	120	☐
Condiments	Gula Malacca	20 - 2 kg bags	27	☐
Condiments	Louisiana Fiery Hot Pepper Sauce	32 - 8 oz bottles	76	☐
Condiments	Louisiana Hot Spiced Okra	24 - 8 oz jars	4	☐
Condiments	Northwoods Cranberry Sauce	12 - 12 oz jars	6	☐
Condiments	Original Frankfurter grüne Soße	12 boxes	32	☐
Condiments	Sirop d'érable	24 - 500 ml bottles	113	☐
Condiments	Vegie-spread	15 - 625 g jars	24	☐
Confections	Chocolade	10 pkgs.	15	☐
Confections	Gumbär Gummibärchen	100 - 250 g bags	15	☐
Confections	Maxilaku	24 - 50 g pkgs	10	☐
Confections	NuNuCa Nuß-Nougat-Creme	20 - 450 g glasses	76	☐
Confections	Pavlova	32 - 500 g boxes	29	☐
Confections	Schoggi Schokolade	100 - 100 g pieces	49	☐
Confections	Scottish Longbreads	10 boxes x 8 pieces	6	☐
Confections	Sir Rodney's Marmalade	30 gift boxes	40	☐
Confections	Sir Rodney's Scones	24 pkgs. x 4 pieces	3	☐
Confections	Tarte au sucre	48 pies	17	☐
Confections	Teatime Chocolate Biscuits	10 boxes x 12 pieces	25	☐
Confections	Valkoinen suklaa	12 - 100 g bars	65	☐
Confections	Zaanse koeken	10 - 4 oz boxes	36	☐

Ar trebui să vă fie clar acum cât de utile sunt interogările. Vizualizare Datalist nu este, totuși, prea utilă pentru utilizatorul final, iar aici intervi rapoartele. Folosind un raport puteți realiza o așezare în pagină car ușurează vizualizarea datelor și care este mai utilă utilizatorului. Interogăril adună la un loc datele din unul sau mai multe tabele, iar rapoartele ofer un format util pentru vizualizarea datelor. Figura 2.17 demonstrează cu un raport poate însufleți o interogare.

FIGURA 2.17

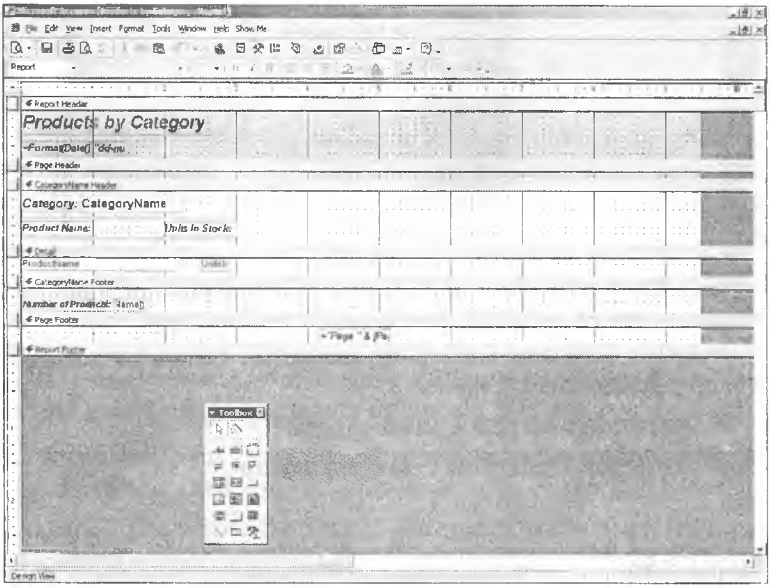
Raportul Products by Category în vizualizarea Print Preview.



Structura unui raport este ceva mai complicată decât a unui tabel sau a unei interogări. Figura 2.18 prezintă raportul Products by Category în vizualizarea Design.

Detaliile despre construirea rapoartelor sunt discutate în capitolul 12.

FIGURA 2.18
Raportul Products by Category în vizualizarea Design.



Formulare

Ultimele obiecte ale bazelor de date din Access pe care le trecem în revistă în această prezentare generală sunt formularele. Ați văzut unde sunt stocate datele (tabelele), modul în care datele pot fi combinate într-un fomat util (interogările) și modul în care datele pot fi prezentate într-un format util (rapoartele). Dar cum pot fi introduse datele în baza de date? Pot fi introduse în două moduri:

- Folosind vizualizarea Table Datalist (vedeți figura 2.6);
- Folosind introducerea datelor în formulare.

Deși puteți introduce, modifica și șterge datele în vizualizarea Datalist, acest lucru nu este foarte util. Formularele pentru introducerea datelor oferă o cale prietenoasă de interacțiune cu datele față de utilizator. Ca și în cazul rapoartelor, structura formularelor poate fi foarte complexă. Deși aspectele importante sunt prezentate în acest capitol, detaliile despre construirea formularelor sunt explicate în capitolul 11.

Figura 2.19 prezintă formularele definite în baza de date Northwind Traders.

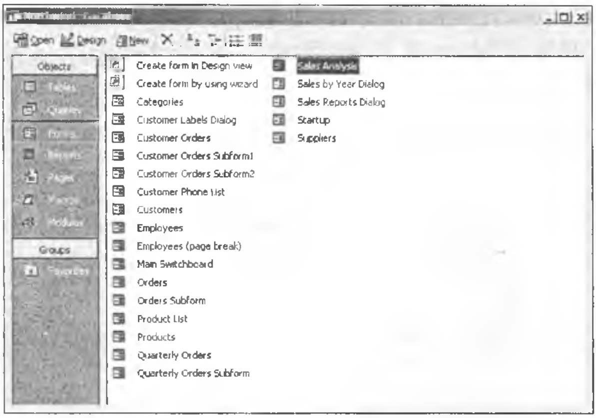


FIGURA 2.19
Formularele definite în baza de date Northwind Traders.

Tabelul Products a fost și va fi discutat în mai multe contexte. Formularul Products oferă o bună ilustrare a funcționării formularelor, dar și a avantajelor acestora (vedeți figura 2.20). Să ne amintim relațiile în care este implicat tabelul Products:

- Un produs aparține unei categorii.
- Un produs este livrat de un furnizor.

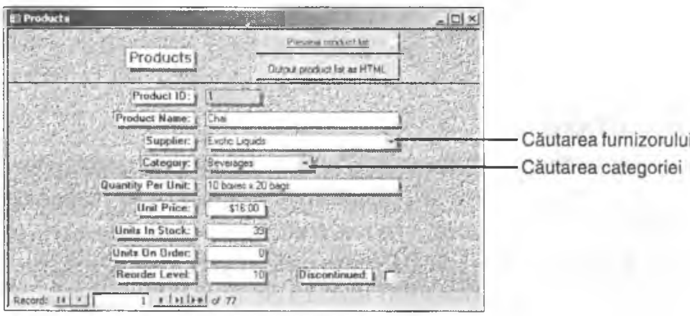
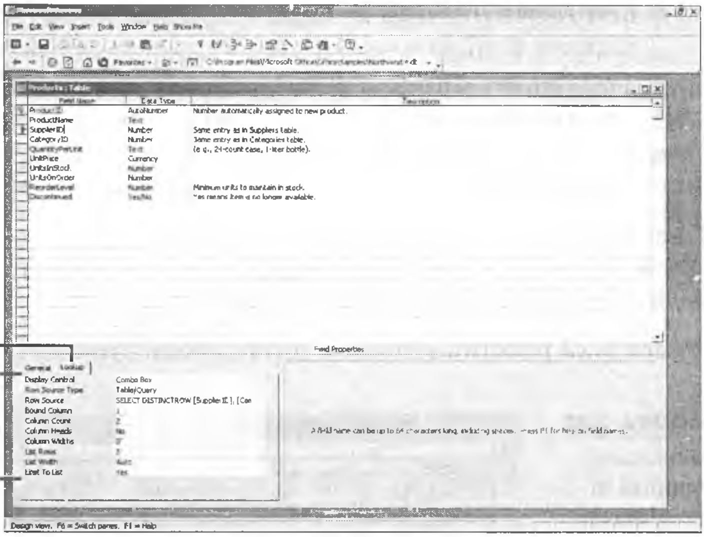


FIGURA 2.20
Formularul Products oferă o modalitate de introducere a datelor prietenoasă față de utilizator pentru un anumit produs.

Doarece tabelul Products este implicat în două relații, există două căutări (***lookup***) pentru acest tabel. Acestea se referă la câmpurile SupplierID (identificatorul furnizorului) și CategoryID (identificatorul categoriei). Dacă vă amintiți cele discutate la începutul capitolului aceste câmpuri cu cheie externă trimit la tabelele Supplier (furnizor) și Category (categorie), respectiv. Informații despre modul în care aceste date vor fi afișate într-un formular pot fi găsite în tab-ul Lookup din Table Designer. Tab-ul Lookup pentru câmpul SupplierID din tabelul Products este prezentat în figura 2.21.

FIGURA 2.21

Tab-ul Lookup oferă informații și componente ale interfeței cu utilizatorul pentru suportul construcției formularului.



Căutările sunt mai ușor de folosit, deoarece utilizatorii pot lucra mai bine cu descrieri în loc de valori cheie numerice lipsite de înțeles.

Folosind câmpul SupplierID ca exemplu, în continuare sunt prezentate informațiile oferite de tab-ul Lookup:

- ***Display Control (controlul afișat)*** – Elementul de interfață cu utilizatorul (casetă combinată, casetă de tip listă sau casetă de text) folosit pentru a afișa căutarea.
- ***Row Source Type (tipul de rând sursă)*** – O căutare își poate extrage datele dintr-o interogare, din valori statice sau din câmpurile unui alt tabel. În acest caz, câmpurile SupplierID și companyname (numele firmei) sunt interogate din tabelul cu furnizori ca sursă de date pentru căutare.
- ***Row Source (rândul sursă)*** – Interogarea, lista statică sau lista de câmpuri folosită pentru a popula căutarea.
- ***Bound Column (coloana de legătură)*** – Valoarea coloanei înscrisă în câmpul respectiv. În acest caz, valoarea SupplierID din tabelul Supplier este înscrisă în câmpul SupplierID din tabelul Products. Câmpul SupplierID este primul câmp din interogare; de aceea, coloana de legătură pentru căutare este setată la 1.
- ***Column Count (numărul de coloane)*** – Specifică numărul de coloane care vor fi afișate.
- ***Column Heads (titlurile coloanelor)*** – Specifică dacă sunt afișate titlurile coloanelor.
- ***Column Widths (lățimea coloanelor)*** – Specifică lățimea fiecărei coloane. În acest caz, prima coloană, SupplierID, are lățimea 0 și de aceea nu este afișată. Singura coloană afișată este coloana 2, Company Name (numele firmei).

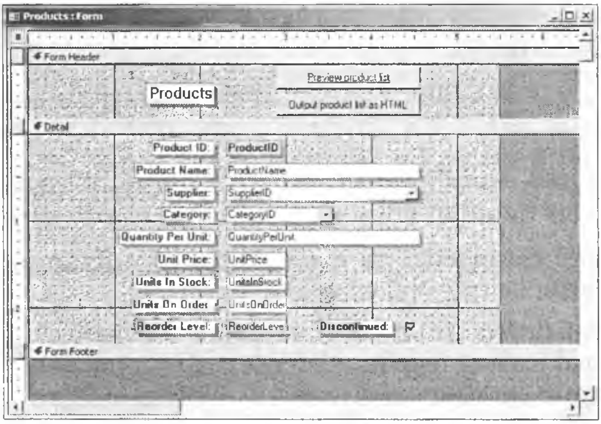
- ***List Rows (rânduri în listă)*** – Specifică numărul maxim de rânduri care vor fi afișate.
- ***List Width (lățimea listei)*** – Specifică lățimea porțiunii din listă destinată unei casete combinate și a unei casete de tip listă.
- ***Limit To List (limită a listei)*** – Dacă este setată pe yes (da), numai valorile care apar în listă pot fi introduse.

Trecerea de la o vizualizare la alta într-un formular este similară cu cea din tabele, interogări și rapoarte: executați clic pe butonul View pentru a trece de la vizualizarea Design la vizualizarea Form și invers.

Figura 2.22 prezintă structura formularului Products.

FIGURA 2.22

Formularul Products în vizualizarea Design.



Rezumat

În concluzie, ați văzut până acum cele mai importante elemente dintr-o bază de date Access, și anume:

- Tabele
- Relații
- Interogări
- Rapoarte
- Formulare

Datele sunt stocate în tabele, iar relațiile definesc modul de asociere a diferitelor tabele. Interogările combină tabelele și relațiile dintre ele pentru a face datele utile, în timp ce rapoartele aduc datele la viață. În sfârșit, formularele permit utilizatorilor să interacționeze cu datele.

Examinând o bază de date din Access, puteți vedea produsul finit, și probabil, veți înțelege modul cum funcționează împreună toate elementele. Acest capitol vă oferă o privire generală asupra pădurii arătându-vă rapid copacii. Dacă înțelegeți imaginea de ansamblu, vă va fi mult mai ușor să înțelegeți detaliile care vor fi prezentate în celelalte capitole.

Acum că ați înțeles ce este o bază de date și cum sunt bazele de date implementate în Access, este timpul să aplicați aceste cunoștințe prin crearea unei baze de date proprii. Următoarele trei capitole au ca scop proiectarea și construirea unei baze de date cu date temporale și pentru facturare. Pe măsură ce sunteți condus în procesul de proiectare și construire a bazei de date Time Entry and Billing, nu uitați baza de date Northwind Traders, deoarece veți aplica aceleași principii – singura diferență este problema pe care o rezolvați.

PARTEA A II-A

PROIECTAREA ȘI CONSTRUIREA PRIMEI BAZE DE DATE



3

INTRODUCERE ÎN PROIECTAREA BAZELOR DE DATE

În acest capitol:

- Proiectul dumneavoastră: baza de date Time Entry and Billing
- Determinați cerințele privind baza de date
- Introducere în modelarea bazelor de date
- Modelul bazei de date Time Entry and Billing
- Rezumat

Partea I v-a oferit elementele fundamentale pentru a înțelege ce este o bază de date și cum este ea implementată în Microsoft Access. În partea a doua, începând cu acest capitol, ne vom concentra eforturile să aplicăm ceea ce am învățat în partea I. Înainte de a putea construi o bază de date, va trebui mai întâi să o proiectați. Înainte de a putea să o proiectați, va trebui să înțelegeți problema economică pe care o aveți de rezolvat. Uneori, procesul de proiectare și construire a unei baze de date este simplu; alteori, acest proces poate fi o adevărată provocare. Printre factorii care pot determina dificultatea sa sunt:

- Modul în care ați înțeles activitatea firmei.
- Aptitudinile dumneavoastră analitice.
- Numărul de baze de date pe care le-ați mai dezvoltat.
- Cantitatea de răbdare pe care o aveți la dispoziție.

Dintre toate aceste aptitudini, răbdarea este probabil cea mai importantă și mai greu de obținut. În timp, veți cunoaște bine activitatea firmei, vă veți dezvolta capacitatea analitică și veți construi destule baze de date. Bazele de date nu sunt niciodată complet dezvoltate de la prima încercare. La sfârșitul celei de-a goua încercări, scopul dumneavoastră va fi aproape îndeplinit. Cu puțin noroc, totul va fi gata la al treilea atac. Mai mult decât probabil este, totuși, că vor fi necesare patru încercări pentru a realiza un proiect inițial solid. Numai dacă aveți răbdare să ajungeți la sfârșitul fiecărei încercări, vă veți putea îndeplini scopul de a proiecta o bază de date solidă.

Subiectul proiectării unei baze de date poate fi foarte alunecos într-o lume a teoriilor complexe. Teoriile privind proiectarea par să nu se aplice în lumea bazelor de date – nu este adevărat. La începutul unei cărți introductive în bazele de date, totuși, aproximativ două treimi din ele sunt inutile. Acest capitol și partea a II-a sunt o combinație formată din teorie (o treime) și muncă grea (două treimi).

Vom învăța făcând. Să trecem la treabă!

Proiectul dumneavoastră: o bază de date cu elemente temporale și de facturare

Să presupunem că ați fost angajat de firma de avocatură Dewey, Cheatem, and Howe pentru a construi o bază de date cu elemente temporale și de facturare. În afară de câteva bancuri despre avocați (precum „De ce rechinii nu mănâncă avocați? Din solidaritate profesională.") nu știți prea multe detalii despre activitatea dintr-o firmă de avocatură. Pe baza numelui bazei de date, știți, totuși, două lucruri:

- Sistemul presupune colectarea de date temporale.
- Sistemul presupune crearea facturilor.

Acest lucru ar putea părea extrem de simplu, dar uneori ar putea fi tot ce aveți pentru început. Așa cum veți învăța, obținerea unei informații poate duce la obținerea a încă una, două sau mai multe informații suplimentare ș.a.m.d. Poate că informațiile cele mai importante pe care le puteți deține la începutul unui proiect sunt locul și modul în care veți găsi informațiile necesare pentru construirea unei baze de date. Acest subiect face obiectul secțiunii următoare.

Determinați cerințele unei baze de date

Cum veți proceda pentru a decide ce cerințe trebuie să îndeplinească baza de date? Elementul cheie pentru a decide acest lucru este reprezentat de faptul că știți unde să găsiți informațiile. În ciuda marilor diferențe dintre firme și departamentele acestora, sursele de informații sunt aceleași. Pe măsură ce veți dezvolta diferite sisteme de baze de date, veți vedea că procesul de adunare a datelor privind cerințele care trebuie îndeplinite este foarte repetabil. Printre multele surse de informare în acest sens, cele mai obișnuite sunt interviurile, formularele folosite de firmă și sistemele existente.

Interviurile

Cu toate sistemele de calcul utilizate azi de firme, tot oamenii sunt cei care iau decizii și fac lucrurile să se întâmple. Nu trebuie să subestimați rolul comunicării și – poate mai important – nu subestimați necesitatea de a asculta ce au de spus angajații unei firme. Poate că acești oameni nu pot vorbi în termenii tehnici ai bazelor de date, dar, din nou, aici interveniți dumneavoastră – pentru a aduna și „traduce" aceste informații.

Prin intermediul interviurilor, veți putea să învățați cum circulă informațiile dintr-o firmă. Veți învăța și pe care anume elemente de date se bazează oamenii și ce decizii iau pornind de la ele.

Indiciu

Sunteți pe cale să reconstituiți un puzzle.

Când luați interviurile, nu vă limitați numai la personalul de nivel superior și la angajații următorului nivel ierarhic. Fiecare nivel al organizației contribuie cu câte o piesă la acest puzzle. Fiecare este la fel de important, pentru că, fără toate piesele nu veți putea niciodată avea imaginea completă asupra organizației, deci nici asupra cerințelor pe care trebuie să le îndeplinească baza de date pe care o construiți.

NECESITĂȚI STRATEGICE VERSUS NECESITĂȚI TACTICE

Deciziile pe care le ia o firmă sunt de două tipuri:

- Strategice;
- Tactice.

Deciziile strategice sunt acelea care se referă la aspecte precum adoptarea unei anumite linii de afaceri, începerea sau terminarea unei anumite linii de produse etc. Deciziile strategice afectează firma ca întreg. Din punct de vedere temporal, orizontul planificării strategice este de obicei mai mare de șase luni și poate implica perioade de până la cinci ani. Este posibil să cunoașteți planurile companiei pentru trei, patru sau cinci ani, iar dacă vi se cere să proiectați o bază de date pentru firmă, trebui să cunoșteți planurile ei strategice. Este probabil ca baza de date să trebuiască să susțină aceste planuri, furnizând informații pentru luarea deciziilor.

În timp ce deciziile strategice sunt pe termen lung, ***deciziile tactice*** sunt pe termen scurt și sunt mai detaliate. Deciziile strategice se concentrează pe ceea ce o firmă dorește să facă; în contrast, deciziile tactice se concentrează pe modul în care aceste lucruri vor fi realizate. De exemplu, o decizie strategică ar putea fi creșterea operațiilor în următorii doi ani. Pentru a alimenta această creștere, ar fi necesară o creștere de 10% a veniturilor firmei. Deciziile tactice care susțin această strategie ar putea fi:

- Creșterea timpului de plată a datoriilor pe termen scurt de la 30 la 60 de zile;
- Solicitarea plății facturilor de către clicenți în 10 zile în loc de 30 de zile;
- Găsirea unor noi resurse de finanțare bancară.

Poate vă întrebați ce legătură au toate acestea cu dezvoltarea bazei de date. Totul! Datele sunt sângele dătător de viață al unei firme! Fără date, o companie nu poate lua decizii. Pentru a lua deciziile tactice despre care am vorbit anterior, o bază de date trebuie să ofere datele solicitate într-un format utilizabil. Prin interviuri, poți afla multe despre ceea ce o firmă dorește să realizeze și, între timp, vei afla care sunt cerințele pe care trebuie să le îndeplinească baza de date.

Formulare și documente ale firmei

Interviurile pot oferi informații de nivel înalt și orientate spre detalii. Examinarea și analizarea formularelor firmei este punctul de început pentru a pătrunde în cele mai mici detalii. Iată câteva exemple de formulare ale unei firme:

- Facturi;
- Rapoarte;
- Documente de expediere a mărfii;
- Fișe de pontaj;
- Sondaje privind serviciile față de clienți;
- Orice tip de hârtii folosite de angajații firmei.

Pe parcursul interviului, poți descoperi că este extrem de important pentru companie să stocheze informații despre utilizatori. Prin analiza și examinarea formularelor și documentelor firmei, poți descoperi informațiile specifice despre clienți care trebuie colectate. Alt exemplu sunt foile de pontaj. În proiectul curent, baza de date privind evoluția în timp și facturarea, știi că trebuie să urmărești destul de detaliat evoluția în timp a lucrurilor. O foaie de pontaj tipică ar putea conține următoarele informații:

- Numele angajatului;
- Data;
- Ora la care începe lucrul;
- Ora la care termină lucrul;
- Activitatea desfășurată;
- Proiectul;
- Clientul.

Examinând acest exemplu privind foile de pontaj, devine mai clar ce informații trebuie să conțină baza de date. Capitolul 4 se concentrează asupra modului de organizare a informațiilor. În acest punct, ceea ce este important este locul în care găsiți informațiile.

Indiciu

Este momentul să vă opriți și să respirați adânc. Este important să vedeți cum funcționează împreună interviurile și formularele firmei. Este o rețea fără asemănare. Într-o anumită măsură, interviurile pot oferi informații și despre detalii. Totuși cea mai mare parte dintre detalii este oferită de analiza formularelor și documentelor firmei.

Sistemele existente

Este foarte probabil ca scopul proiectului dumneavoastră să fie înlocuirea unui sistem existent. Poate că dezvoltarea firmei depășește capacitățile sistemului existent. Sau poate că designul bazei de date existente nu îndeplinește cerințele tot mai mari ale firmei. Pentru a înțelege cerințele de design ale bazei de date, va trebui să înțelegeți sistemul curent. Mai important, trebui să înțelegeți care sunt neajunsurile acestuia. Dacă nu cunoașteți și nu înțelegeți problemele legate de sistemul existent, probabilitatea de a perpetua aceste proobleme și în noul sistem crește dramatic.

Indiciu

Din multe puncte de vedere, componentele interfeței cu utilizatorul ale unui sistem existent pot fi considerate documentele și formularele firmei. De exemplu, poate că nu există fizic bucata de hârtie care conține foaia de pontaj, ci probabil că angajații introduc informațiile respective folosind un ecran special. Ca dezvoltator al bazei de date, va trebui să știți că nu există o linie de demarcație clară între sursele disparate de informații. Mai mult, nu există nici o ordine stabilită de trecere în revistă a diferitelor elemente. În final, este foarte posibil să trebuiască să revizitezi aceleași surse de informații. De exemplu, după ce ați analizat un anumit document, va trebui să reveniți la intervievarea angajaților pentru a obține alte clarificări privind cerințele asupra bazei de date.

Vă amintiți mantra răbdării? Procesul de colectare a cerințelor privind baza de date ar putea să vă pară nesfârșit. Unii oameni cred că sistemele informatice și bazele de date nu sunt niciodată complete, deoarece mediul de afaceri evoluează și se schimbă în continuu. Într-un anumit moment veți fi adunat destule informații pentru a realiza designul inițial al bazei de date, dar numai realizarea designului inițial nu înseamnă că ați încheiat procesul de colectare a informațiilor privind cerințele asupra acesteia. Nu uitați, va trebui să repetați procesul de două, trei sau patru ori înainte de a obține primul design al bazei de date. Pe măsură ce acumulați experiență, vă veți familiariza cu toate aceste lucruri.

În acest punct, știți unde puteți găsi sursele de informații pentru cerințele asupra bazei de date. Acum întrebarea este cum să alcătuiești un design inițial. Răspunsul este un model al bazei de date. Procesul de creare a primului dumneavoastră model al unei baze de date este scopul secțiunii următoare.

Introducere în modelarea bazelor de date

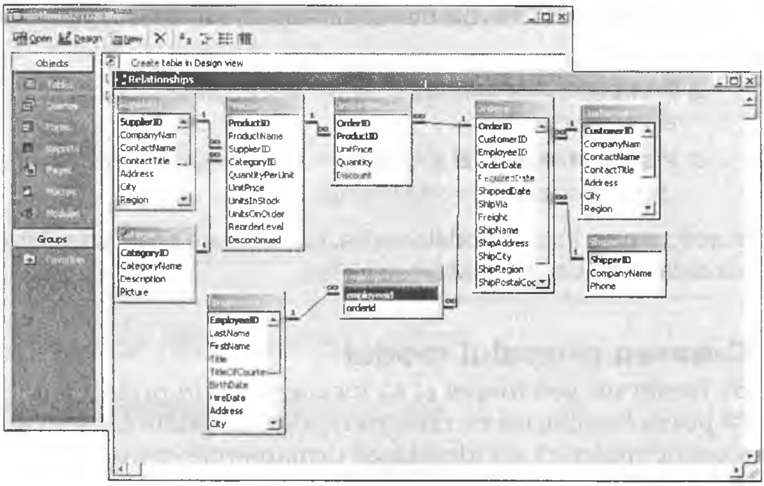
Se spune că o imagine face cât o mie de cuvinte. Nicăieri nu este mai adevărată această zicală ca în lumea modelării bazelor de date. Așa cum ați văzut, bazele de date sunt mai mult decât grupuri de tabele. Bazele de date grupează entități înrudite ale firmei. Puneți-vă întrebarea ce este mai ușor de realizat: o listă statică a numelor de tabele sau o diagramă care prezintă tabelele și relațiile dintre ele? Indiferent dacă sunteți sau nu o persoană cu aptitudini tehnice, este mai ușor de lucrat cu diagramele. Diagramele permit oamenilor să înțeleagă mai ușor problemele și conceptele dificile.

Să considerăm de exemplu, planurile unei clădiri. Privindu-le, puteți înțelege rapid scopul și structura clădirii. Planurile realizează mult mai mult decât ar putea cuvintele să spere că vor realiza vreodată. Într-un anumit sens, un model al bazei de date este ca un plan al acesteia. Scopul acestei secțiuni este de a vă orienta pe calea creării primului dumneavoastră model al unei baze de date.

Cum arată un model al bazei de date?

În capitolul 2 am făcut o introducere în primul model al unei baze de date. Pentru a recapitula, figura 3.1 ilustrează modelul bazei de date Northwind Traders din Access.

FIGURA 3.1
Vizualizarea Relationships vă permite să vedeți toate tabelele dintr-o bază de date și relațiile dintre ele.



Dacă priviți figura 3.1, veți înțelege imediat scopul bazei de date. Capitolul 2 a studiat acest scop, examinând modelul. Acum vom face o introducere în componentele modelului unei baze de date.

CÂTEVA CUVINTE DESPRE INSTRUMENTELE DE PROIECTARE A BAZELOR DE DATE

Instrumentele de proiectare a bazelor de date pot fi pur și simplu creionul și hârtia. Ele pot fi și foarte complexe, încât să cheltuiți pe ele câteva sute de dolari sau câteva mii de dolari. Sam Snead, unul dintre cei mai mari jucători de golf ai timpului, relatează cum a realizat lovitutura de 76 puncte pe un teren dificil în cartea sa, ***The Game I Love***. Lecția pe care o primim de aici nu se referă la instrumentul pe care îl folosim, ci la eficiența cu care îl folosim. Un desen realizat cu creionul pe un petec de hâtie poate fi la fel de bun precum unul realizat cu un instrument care costă mii de dolari. Bine, poate că am exagerat puțin. Totuși, nu trebuie să faceți o ipotecă pentru a vă procura un instrument bun de proiectare a bazei de date. Pe piață există două astfel de instrumente relativ convenabile:

- ***Microsoft Visio*** – De la preluarea de către Microsoft, Visio a devenit parte a suitei Microsoft Office. Pentru mai multe informații, intrați pe site-ul Microsoft Visio Web, la `http://www.microsoft.com/office/visio/`.
- ***Resolution xCase*** – acesta este un instrument complet pentru proiectarea bazelor de date care funcționează cu SQL Server, Access, Sybase, Informix, DB2 și multe alte baze de date existente pepiață. Pentru mai multe informații, intrați pe site-ul Resolution xCase, la `http://www.xcase.com`.

Desigur, dacă aveți nevoie de instrumente foarte specalizate, poate veți fi interesat de:

- ***Computer Associates ErWin*** – Informații la `http://ca.com/products/alm/erwin.htm`.
- ***Visible Analist DB Engineer*** – Informații la `http://www.visible.com/dataapp/dappprods/vadbe.htm`.

În scopul de a ilustra modelarea bazelor de date și noțiunile înrudite acestei teme, această carte va utiliza Microsoft Visio.

Crearea primului model

Să facem un pas înapoi și să începem cu un proiect gata realizat pentru a vă putea familiariza cu crearea modelului grafic al bazei de date. În această situație ipotetică ați identificat următoarele entități:

- Clienți;
- Comenzi;
- Elemente ale comenzilor.

> **Notă**
>
> Toate conceptele din această secțiune pot fi utilizate cu instrumente simple precum creionul și hârtia. Instrumentele de proiectare grafică automată ușurează o parte din muncă, cum ar fi faptul că se asigură că sunt create chei externe în tabelele copil și descifrează dacă este vorba despre o relație unu la unu sau unu la mulți. Unele instrumente automatizate pot chiar realiza scripturi care creează automat definiții ale bazei de date pornind de la model. Mai mult, unele instrumente sunt suficient de inteligente pentru a sincroniza modelul și baza de date ori de câte ori este modificat modelul. Toate aceste funcții sunt drăguțe, dar nici una dintre ele nu este crucială când este vorba de aplicarea conceptelor de modelare și a notațiilor. Nu uitați, important este violonistul, nu vioara!

Când lansați Visio, sunteți întâmpinat de o casetă de dialog care vă invită să alegeți template-ul (șablonul) de desen potrivit (vedeți figura 3.2).

După ce ați selectat folderul Database, veți vedea șase template-uri (vedeți figura 3.3). Pentru acest model, selectați template-ul Crow's Foot ERD.VST. Dacă vă amintiți cele discutate la începutul acestui capitol, acronimul ERD vine de la Entity Relationship Diagram.

După ce ați deschis șablonul, puteți utiliza imediat Visio Design Surface. Visio Designer este prezentat în figura 3.4.

FIGURA 3.2

Visio poate crea multe tipuri de desene tehnice, inclusiv diagrame ale modelelor de baze de date.

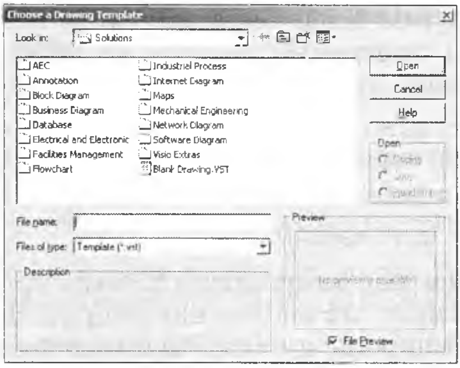


FIGURA 3.3

Visio conține cinci modele de proiecte ale bazei de date. Cel mai comun dintre acestea este Crow's Foot ERD.

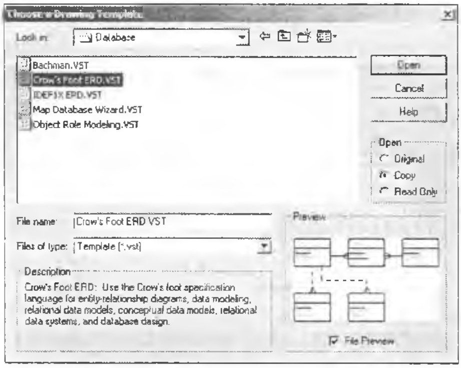
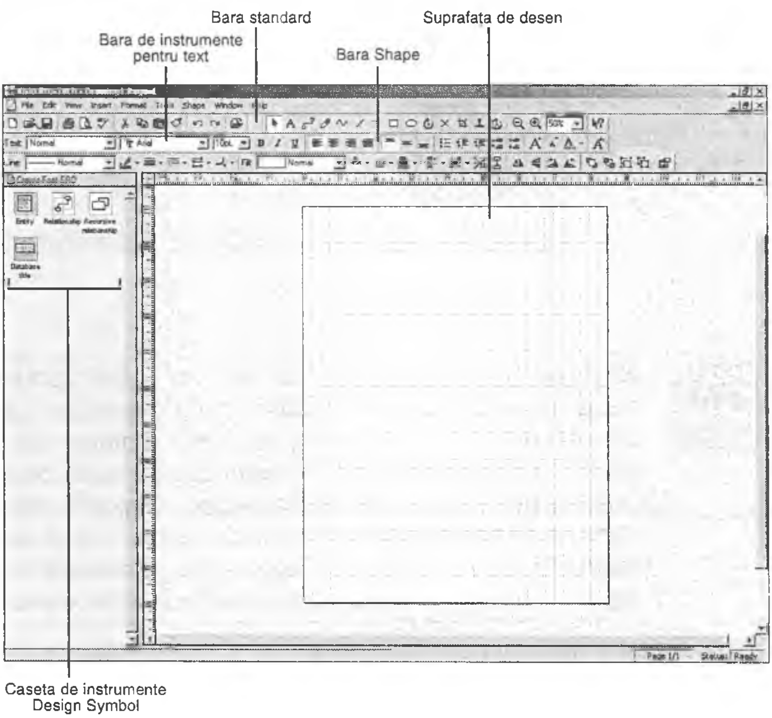


FIGURA 3.4

Visio Design Surface este lansat sub forma unei pagini albe. Aveți la dispoziție multe bare de instrumente pentru a crea absolut orice tip de formă sau orice control cu orice atribute.



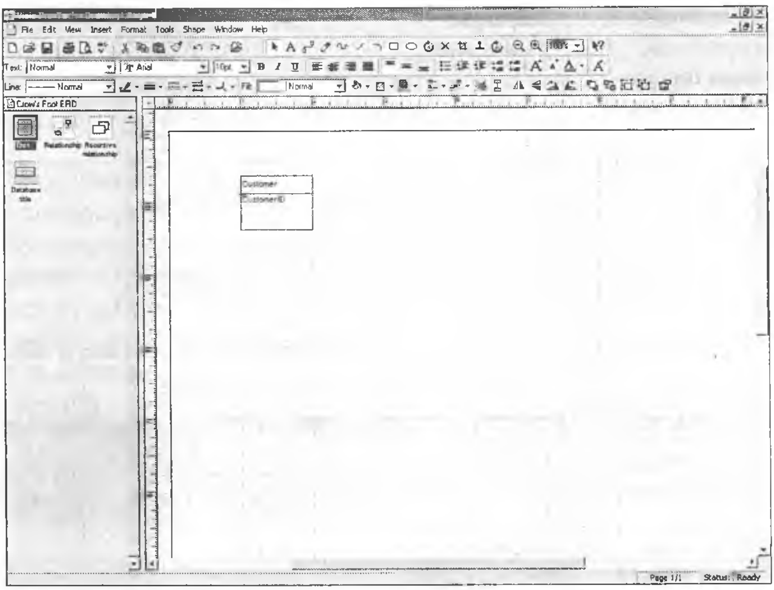
Pornind de la o suprafață de desen liberă, sunteți gata să creați primul grafic. Procesul implică tragerea simbolurilor de desen din caseta de instrumente Design Symbol în suprafața de desen. În acest proces veți parcurge pașii următori:

1. Plasați o entitate pe suprafața de desen.
2. Adăugați atribute acestei entități (nume, câmpuri și chei).
3. Specificați relațiile între entități.

Pentru a înțelege primii doi pași, priviți figura 3.5. Aceasta prezintă entitatea Customer cu un câmp definit. Vor mai fi și alte câmpuri, desigur. Ordinea în care trebuie introduse câmpurile și entitățile în care vor fi introduse acestea sunt discutate în capitolul 4.

FIGURA 3.5

Entitatea Customer are două atribute: numele entității și câmpul cheie primară.



Indiciu

Să ne oprim puțin pentru a relua niște concepte importante. Deși subietul alegerii câmpurilor și a entitățiior în care acestea trebuie incluse este abordat abia în capitolul următor, unele câmpuri trebuie deja adăugate chiar în acest proces. Rolul acestor câmpuri este de a susține diferitele relații dintre entități. Este vorba despre câmpurile cheie primară și cheie externă (străină). Aceste concepte au fost introduse în capitolul 1 și exemplificate în capitolul 2. Dacă trebuie să revedeți rapid aceste noțiuni, este mai bine să o faceți acum, înainte de a trece mai departe.

După ce ați văzut cum arată entitățile în Visio, să examinăm câteva momente notația unei entități (vedeți figura 3.5).

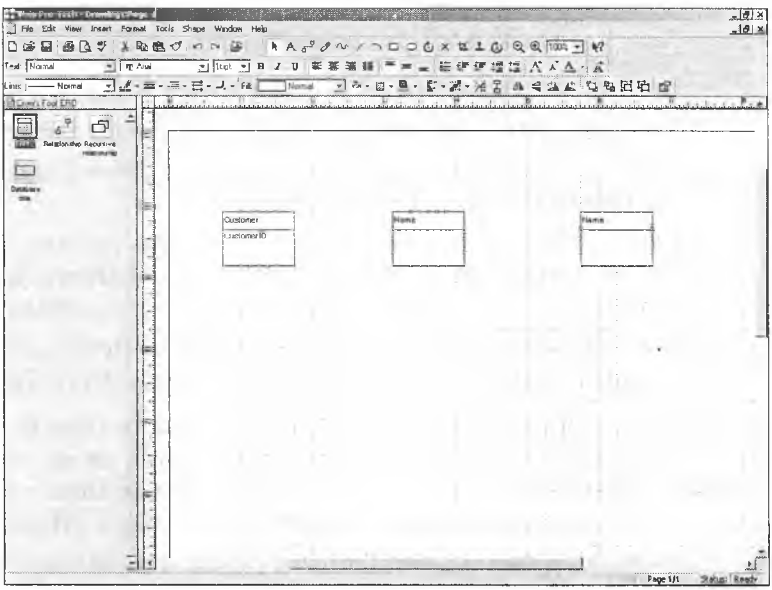
Tipul diagramei create este ***Entity Relationship Diagram*** (***ERD***) – diagrama relațiilor între entități. O entitate din ERD este alcătuită dintr-un dreptunghi, numele entității și lista câmpurilor. După cum vedeți, nu este nevoie de un design special pentru a crea diagramele. Așa cum vom arăta în cele ce urmează, totuși, instrumentele de design fac munca mult mai ușoară.

Crearea entităților în Visio este foarte simplă. Pașii următori vă ghidează prin procesul creării entităților Orders și Order Line Items:

1. Selectați simbolul entității din caseta de instrumente Design Symbol.
2. Ținând apăsat butonul stâng al mouse-ului, trageți simbolul pe suprafața de desen.
3. După poziționarea sa pe suprafața de desen, eliberați butonul mouse-ului. Va fi creată astfel o nouă entitate.
4. Selectați entitatea nou creată din suprafața de desen și apăsați Ctrl+C pentru a copia entitatea în Windows Clipboard.
5. Apăsați Ctrl+V pentru a lipi o nouă entitate pe suprafața de desen. Aceasta va arăta ca în figura 3.6, după ce ați aranjat entitățile.

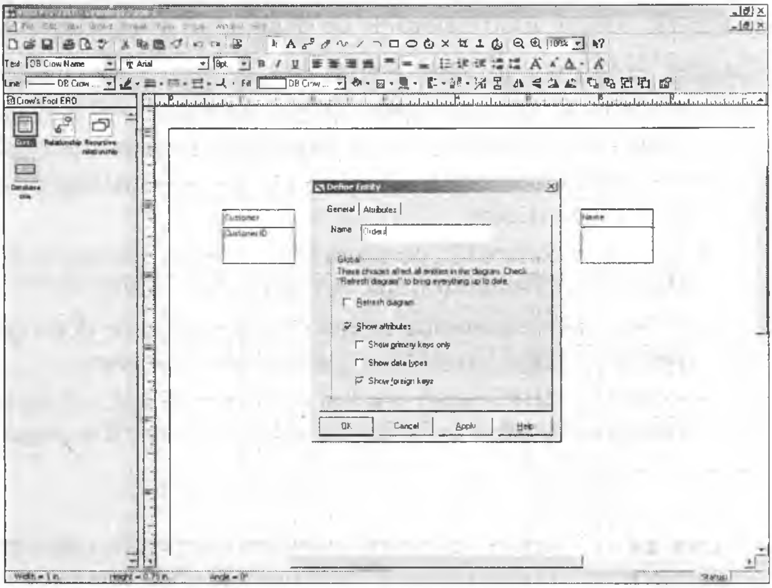
FIGURA 3.6

Cele două entități libere din suprafața de desen reprezintă entitățile Orders și Order Line Items, respectiv.



6. Selectați entitatea din mijloc, executați clic dreapta și selectați Add Attributes din meniul pop-up. Denumiți entitatea Orders și executați clic pe OK. Figura 3.7 prezintă primul tab din caseta de dialog Define Entity.

FIGURA 3.7
Tab-ul General din caseta de dialog Define Entity vă permite să specificați numele entității și diferite atribute de afișare în diagramă.



7. Repetați pasul 6 pentru cealaltă entitate, denumind-o OrderLineItems.
8. Revenind la entitatea Orders, repetați pasul 6, selectând tab-ul Attributes din caseta de dialog Define Entity.
9. Executați clic pe butonul New pentru a crea un nou câmp. Denumiți-l OrderID, specificând că acesta este cheia primară și specificați tipul de date ca Integer. Figura 3.8 prezintă cum va arăta caseta de dialog.
10. Repetați pasul 9 pentru entitatea OrderLineItems, adăugând un câmp numit OrderLineItemID. Diagrama va arăta ca în figura 3.9.

Abia acum începe distracția cu adevărat și abia acum iese în evidență puterea unui instrument de proiectare automată. După ce ați creat entitățile și cheile primare asociate lor, pot fi create relațiile dintre entități. Crearea relațiilor, ca și a entităților, este foarte simplă dacă urmați pașii:

1. Selectați simbolul relațiilor din caseta de instrumente Design Symbol.
2. Trageți simbolul relației pe suprafața de desen. După plasarea pe suprafață, veți vedea linia relației. Înainte de a lăsa linia, asigurați-vă că extremitatea ei din stânga atinge unul dintre marcajele albastre din entitatea Customer. Dacă ați efectuat corect operațiunea, extremitatea stângă va fi roșie când selectați linia relației (vedeți figura 3.10).

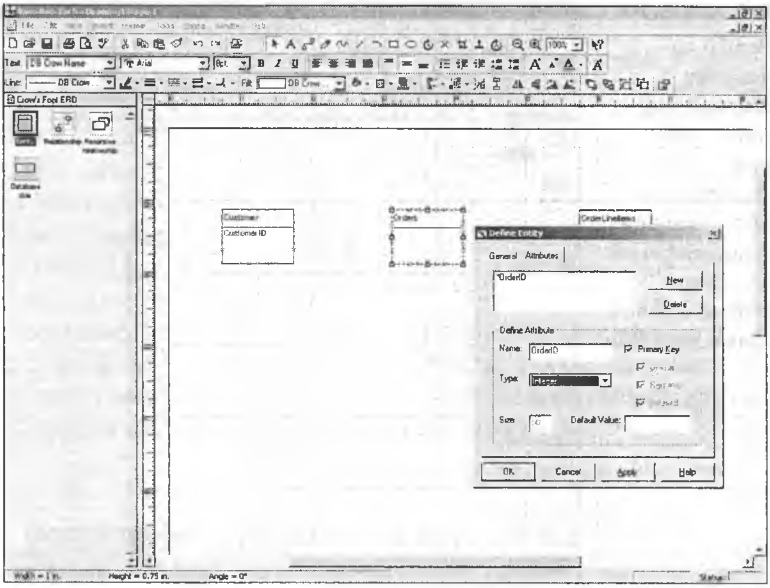


FIGURA 3.8
Tab-ul Entity din caseta de dialog Define Entity vă permite să precizați câmpurile și atributele corespunzătoare lor pentru orice entitate.

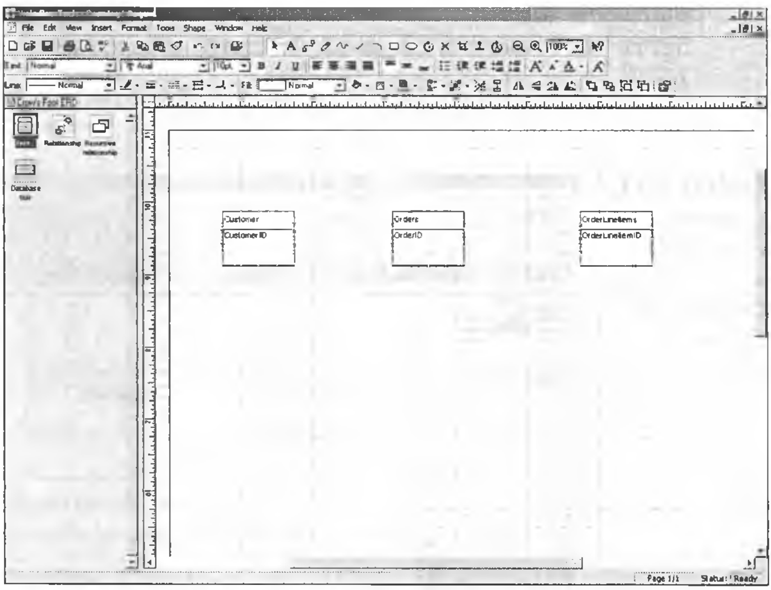
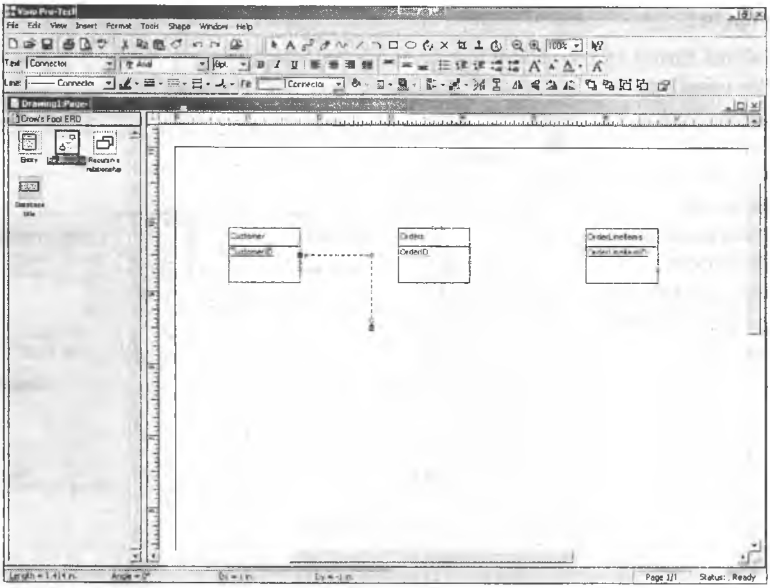


FIGURA 3.9
Modelul ERD cu trei entități, fiecare având definită o cheie primară.

FIGURA 3.10

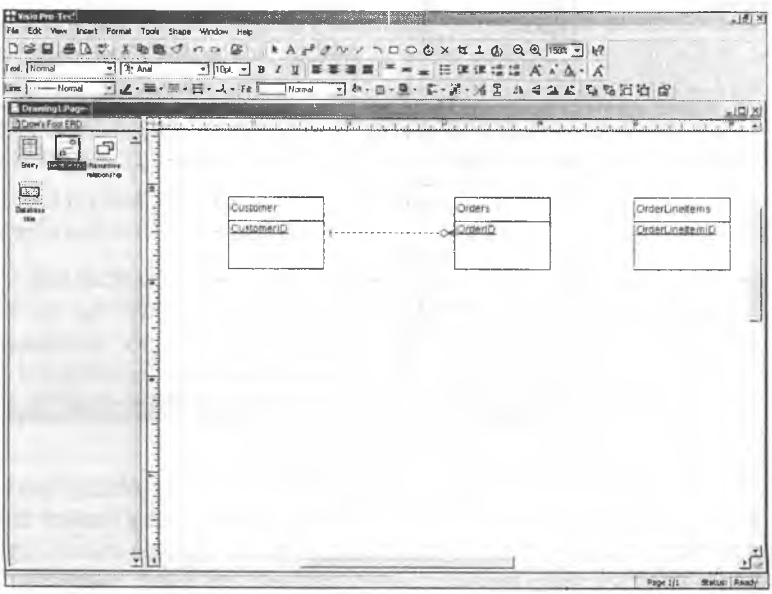
Pentru ca sarcinile automate să funcționeze corect, este important ca fiecare capăt al liniei relației să atingă un marcaj albastru atașat entității.



3. Pasul următor este de a atașa entitatea Orders la capătul din dreapta al liniei relației. Executați clic pe el și trageți-l spre unul dintre marcajele albastre ale entității. Diagrama va arăta precum cea din figura 3.11. pentru ca ea să fie clară, a fost mărită la aproximativ 150%.

FIGURA 3.11

Linia relației conectează acum entitățile Customer și Orders.



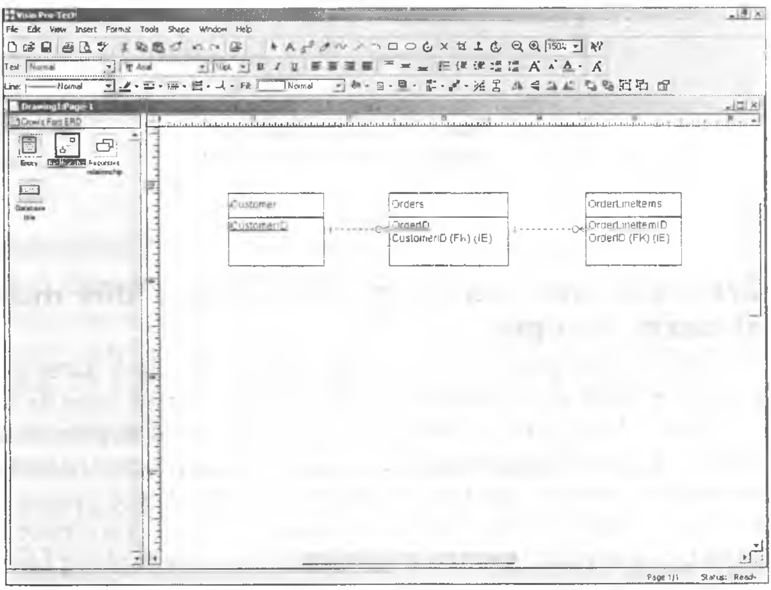
4. Repetați pasul 3 pentru o relație între entitățile Orders și OrderLineItem.

S-ar putea să vă întrebați unde este automatizarea. Până la urmă, dumneavoastră ați făcut toată treaba. Așteptarea a luat sfârșit: acum veți vedea rezultatele muncii dumneavoastră îndârjite!

Din meniul Shape, selectați Update Foreign Keys. Voilà! Pe baza relațiilor, instrumentul a știut ce chei străine trebuie definite. De exemplu, pentru ca entitatea Order să suporte o cheie străină de la CustomerID, trebuie definit un câmp CustomerID. Așa cum veți vedea în capitolul următor, este opțional să păstrați unele câmpuri în ambele entități, dar câmpurile de la acest nivel nu sunt opționale. De aceea discutăm aici despre aceste câmpuri. Mai simplu formulat, dacă nu aveți câmpuri cheie externă, nu puteți avea relații. De fapt, relațiile sunt esența unei baze de date relaționale. Dacă ați parcurs corect pașii, diagrama dumneavoastră va arăta ca în figura 3.12.

FIGURA 3.12

Modelul bazei de date este complet acum, conținând entități, chei externe și relații.



Indiciu

Totuși, de ce se numește „crow's foot" (piciorul ciorii)? Figura 3.13 explică de ce modelul are acest nume aparent demodat.

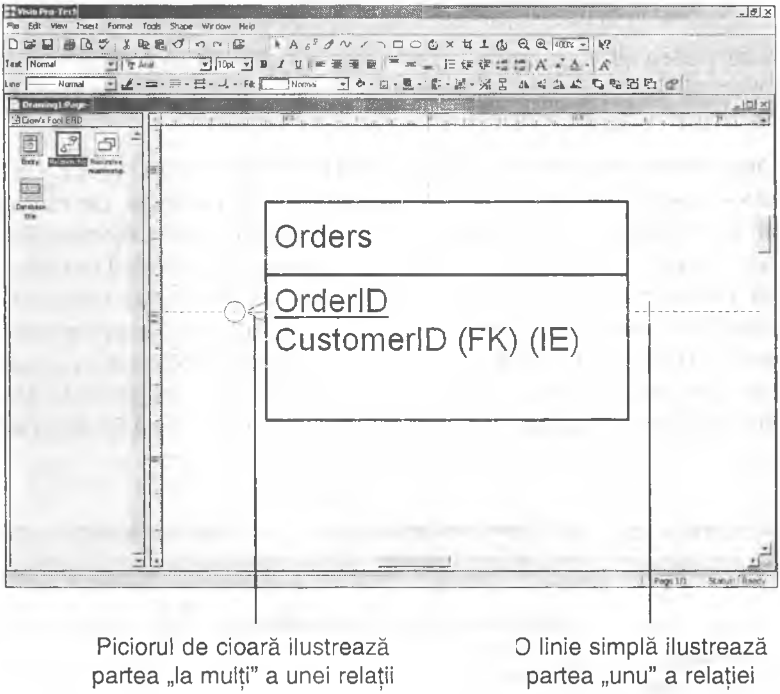


FIGURA 3.13 Piciorul de cioară ilustrează partea „la mulți" a unei relații. O linie simplă ilustrează partea „unu" a relației.

Crearea unei baze de date reale din modelul logic al bazei de date

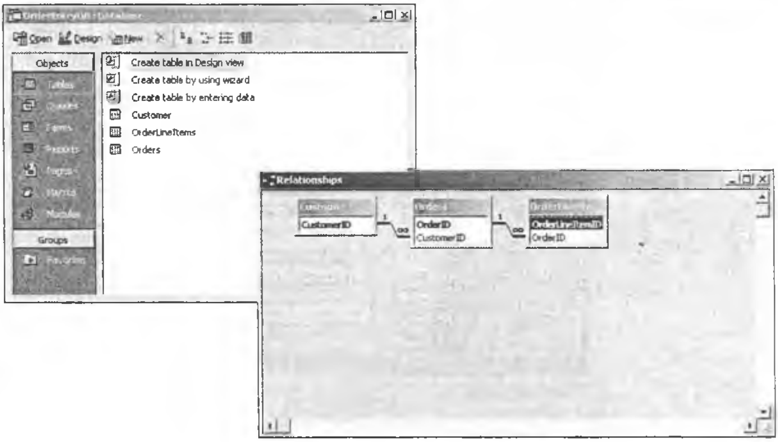
Un instrument automat nu este de prea mare folos dacă nu poate suporta un model logic al bazei de date – adică planul unei baze de date – și crearea unei baze de date reale. Iată de ce proiectarea și implementarea bazelor de date se realizează împreună. Lucrând cu modelele, aveți o șansă de a vizualiza conceptele și ideile înainte de implementare. Acest proces este de departe mult mai eficient decât utilizare unuia de tipul „încercare și eroare" (pe o bază de date reală, pentru a vedea dacă lucrurile merg bine).

Visio și alte instrumente pentru proiectarea bazelor de date au facilități de creare a bazelor de date reale (fizice) din modele logice ale bazelor de date. În cazul programului Visio, există capacitatea de a crea o bază de date în Access, iar procesul este simplu, deoarece este ajutat de o aplicație wizard. Selectați Tools, Macro, Database, Create Database Wizard pentru a lansa aplicația wizard de creare a bazei de date.

> **Atenție** Asigurați-vă că nu există erori ERD. De exemplu, o relație care nu este conectată duce la o eroare ERD. Dacă modelul conține erori, baza de date rezultată va avea aceleași erori. Înainte de a crea baza de date lansând Create Database Wizard pornind de la model, asigurați-vă că ați actualizat cheile externe.

După parcurgerea pașilor din Create Database Wizard, veți obține o nouă bază de date în Access. Diferitele atribute definite în model sunt preluate de baza de date fizică. Figura 3.14 ilustrează rezultatele rulării aplicației wizard Create Database.

FIGURA 3.14
Create Database Wizard a utilizat modelul creat în acest capitol pentru a construi această bază de date în Access.



Crearea unui model logic al bazei de date pornind de la o bază de date fizică

Ce ziceți de situația inversă, când aveți la dispoziție baza de date fizică, dar nu aveți un model? Din fericire, Visio are capacitatea de a efectua și operațiunea inversă, de a crea modelul logic pornind de la baza de date.

Pentru a importa o bază de date, parcurgeți pașii:

1. Din meniul File, selectați New, Database, Map Database Wizard.
2. Indicați locația bazei de date pe care ați creat-o anterior.
3. Alegeți toate entitățile pentru mapare.
4. Creați o diagramă Crow's Foot și lăsați opțiunea Layout așa cum este.
5. Selectați prezentarea tuturor atributelor și a cheilor străine. Lăsați celelalte atribute așa cum sunt.
6. Dați un titlu și o descriere pentru diagrama bazei de date.
7. Executați clic pe butonul Finish.

Figura 3.15 prezintă modul cum va răta modelul rezultat în Visio.

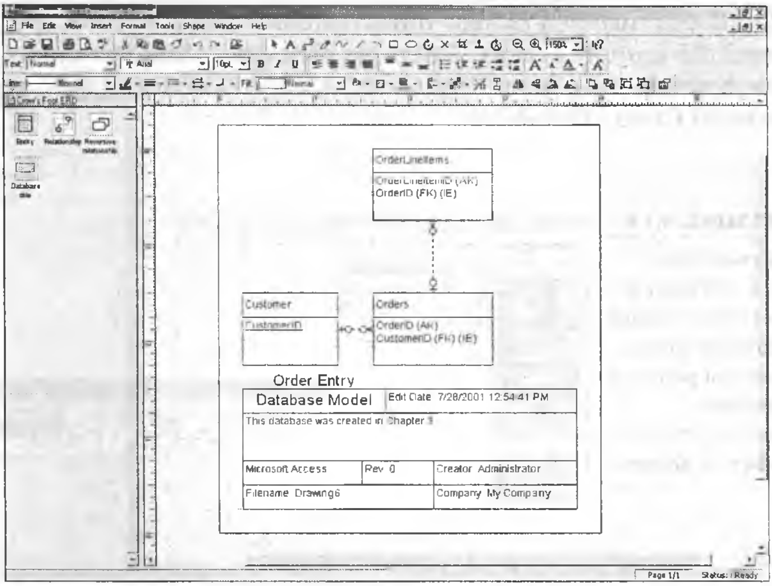


FIGURA 3.15
Map Database Wizard, citind o bază de date existentă, a creat acest model al bazei de date.

Modelarea logică a bazei de date Time Entry and Billing

Acum, după ce ați înțeles ce este modelarea bazelor de date, puteți utiliza tehnicile învățate pentru proiectul curent – baza de date Time Entry and Billing. Desigur, prin interviuri și analiza documentelor și formularelor firmei, ați identificat următoarele entități:

- Clienți;
- Angajați;
- Cazuri;
- Facturi;
- Detalii privind evoluția în timp.

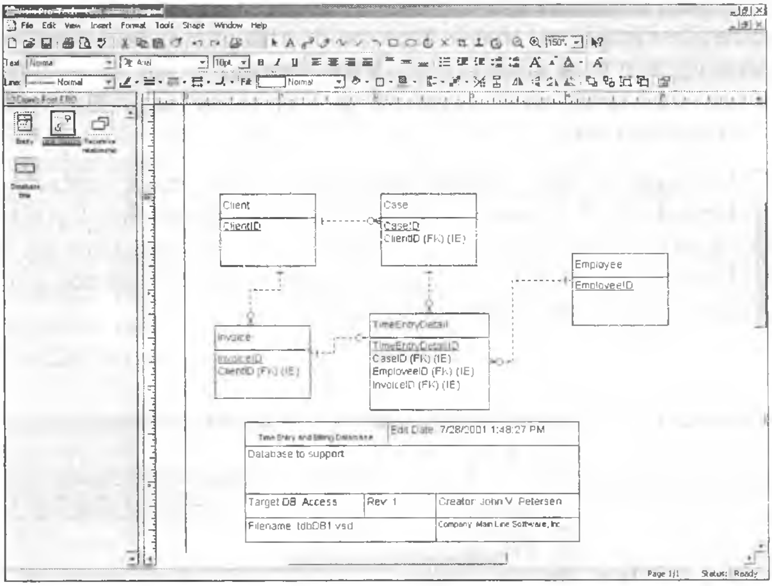
În cursul analizei, ați identificat următoarele relații:

- Un client are unul sau mai multe cazuri;
- Un client are una sau mai multe facturi;
- O factură este alcătuită din unul sau mai multe detalii.
- Timpul este contorizat în fiecare caz.
- Angajații alocă timp fiecărui caz.

Figura 3.16 prezintă modelul bazei de date în acest punct al procesului de dezvoltare. Așa cum veți vedea în curând, există deja câteva asperități care trebuie netezite.

FIGURA 3.16

Versiunea 1 a bazei de date Time Entry and Billing este alcătuită din cinci entități.



Editarea textului

Ați observat forma din diagrama din figura 3.16 care conține informații descriptive despre grafic? Forma este numită titlul bazei de date și este localizată în caseta de instrumente Design Symbol. După ce ați plasat titlul bazei de date pe suprafața de desen, fiecare bloc de text poate fi editat dacă parcurgeți pașii:

1. Executați clic pe caseta cu titlul bazei de date; va fi selectată întreaga formă. Un marcaj verde de selecție va înconjura conturul formei.
2. În acest punct, puteți selecta fiecare dreptunghi conținut în forma cu titlul bazei de date. Pe măsură ce selectați fiecare dreptunghi, conturul său este marcat cu gri.
3. Apăsați tasta F2. Va fi selectat textul, ceea ce vă permite să-l modificați.

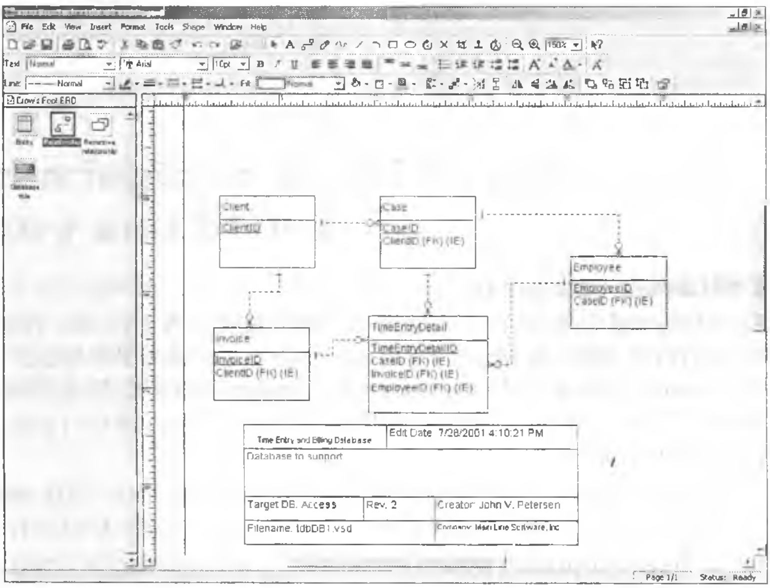
Rafinarea designului

Până acum, proiectul arată bine. Dar este oare complet? Dat fiind că este prima sa variantă, este foarte probabil că nu. Ați observat că relația numărul cinci are un asterisc? Deși, pentru început, această relație poate că nu pare ambiguă, este. Să examinăm ceva mai bine relația „Angajații alocă timp fiecărui caz.”.

Să zicem că în timpul interviurilor ați descoperit că unui angajat îi este atribuit un caz înainte ca el să aloce timp cazului. Designul curent al bazei de date conține o relație directă între cazuri și angajați? Nu, dar conține o relație indirectă. Din entitatea Case, puteți găsi angajați care au alocat timp cazului dacă treceți prin entitatea TimeEntryDetail. Cât timp toți angajații implicați într-un caz au alocat timp cazului, puteți afla angajații care sunt implicați în caz. Dar, pe de altă parte, dacă aveți nevoie de o listă cu angajații care sunt implicați într-un caz și știți că unii dintre ei nu i-au alocat încă timp (nu s-au ocupat încă de cazul respectiv). Pe scurt, înseamnă că aveți ghinion!

Este important de subliniat că relațiile dintre entități trebuie să fie suportate indiferent dacă există instanțieri specifice ale datelor. Cu alte cuvinte, chiar dacă un angajat nu a alocat încă timp unui caz care i-a fost atribuit, el trebuie să apară într-o listă a angajaților care lucrează la cazul respectiv. Figura 3.17 prezintă designul rafinat.

FIGURA 3.17
Versiunea 2 a modelului bazei de date Time Entry and Billing suportă o relație directă între angajați și cazuri.



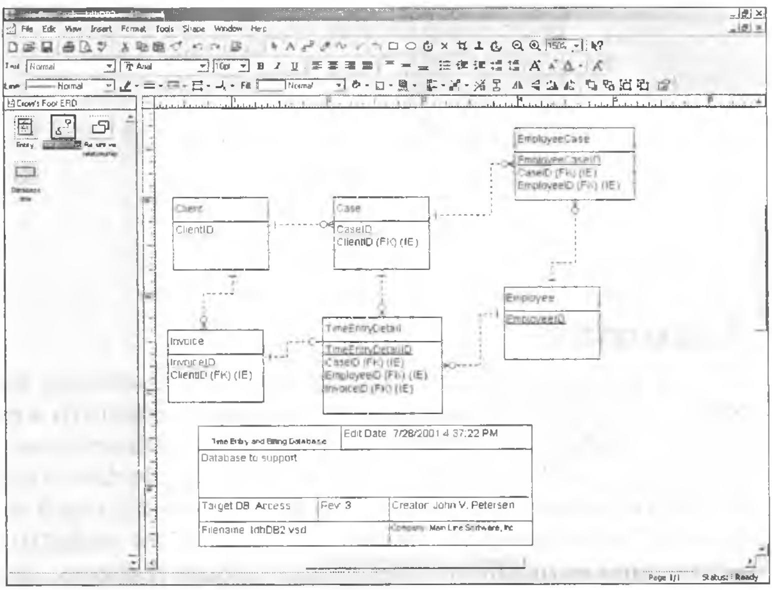
Ați văzut și altă problemă? Pe baza acestui model, când este vorba despre asocierea angajaților cu cazurile, aveți următoarea problemă: de fiecare dată când unui angajat îi este atribuit un nou caz, trebuie creată o nouă înregistrare pentru acel angajat. Acest lucru are ca rezultat înregistrări duble ale angajaților. Astfel, nu va trece mult timp și baza de date va fi aglomerată cu o mulțime de date redundante. Când va trebui să creați rapoarte sau interogări, va fi foarte dificil. Ce s-ar întâmpla dacă un angajat își schimbă numele și lucrează la 10 cazuri? Va trebui să modificați 10

înregistrări ale angajaților. Știți destule despre bazele de date relaționale ca să înțelegeți că acest scenariu este inacceptabil. Unul dintre scopurile principale ale bazelor de date este să asigure consistența datelor, reducând redundanța acestora. Și atunci, cum veți rezolva problema?

Vă amintiți când am discutat despre relațiile unu la mulți în capitolul 2? Este exact situația în care ne aflăm acum. Angajaților li se pot atribui unul sau mai multe cazuri, iar la un caz pot lucra unul sau mai mulți angajați. Ați văzut în capitolul 2 că spargerea unei relații mulți la mulți în două relații unu la mulți redresează situația. În acest caz, pentru a rezolva problema, va fi adăugat modelului un nou tabel intermediar, numit tabel de joncțiune (***jonction table***). Tabelele de joncțiune sunt numite și tabele de rezolvare a relațiilor mulți la mulți (***many-to-many resolver tables***) – dar denumirea nu este nici pe departe atât de importantă ca rolul acestora. Tabelele de joncțiune vă permit să facilitați consistența datelor, reducând redundanța acestora. Figura 3.18 prezintă versiunea 3 a modelului bazei de date, care conține noul tabel de joncțiune și relațiile modificate între entitățile Case și Employee.

FIGURA 3.18

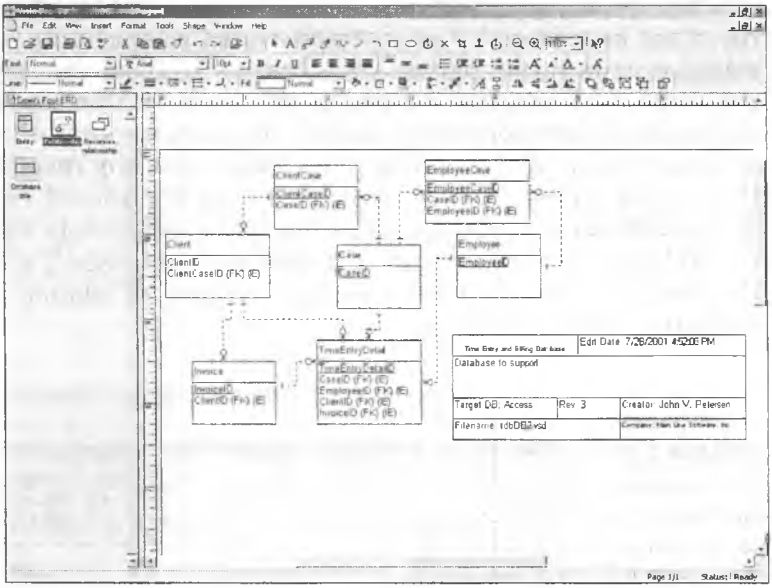
Versiunea 3 a modelului bazei de date Time Entry and Billing suportă o relație mulți la mulți între cazuri și angajați.



Mai trebuie încă să munciți înainte ca primul design al bazei de date să fie considerat complet. Pe măsură ce continuați analiza, veți vedea că unui caz îi pot fi asociați mulți clienți. Datorită acestei sitauții, nu este sufientă asocierea unei înregistrări TimeEntryDetail cu un caz, un angajat și o factură. Deși este adevărat că înregistrarea TimeEntryDetail trebuie să corespundă

unui caz, unei facturi și unui angajat – din cauză că mai mulți clienți pot fi asociați unui caz – entitatea TimeEntryDetail trebuie să aibă ClientID ca și cheie externă. Vedeți figura 3.19 care conține prima versiune completă a bazei de date Time Entry and Billing.

FIGURA 3.19
Versiunea 4 a modelului bazei de date Time Entry and Billing este prima schiță completă.



Rezumat

Într-un scenariu real, veți descoperi că sunt necesare mai multe iterații pentru a obține designul inițial. Timpul necesar pentru a realiza un proiect util depinde de cât de amănunțită este analiza dumneavoastră și de frecvența cu care cereți informații de la utilizatorii bazei de date. Dintre toate sursele de informații disponibile, solicitările de rapoarte ar putea fi cele mai valoroase, deoarece puteți parcurge adesea procesul invers realizării acestora pentru a determina entitățile pe baza cărora trebuie realizate. Desigur, alte surse de informații – interviurile și sistemele existente – sunt și ele importante. În situații diferite, vă puteți baza pe surse diferite.

Rezultatul muncii dumneavoastră de analiză culminează sub forma modelului bazei de date. Cea mai comună și simplă cale de a înțelege tehnica modelării este Entity Relationship Diagram (ERD). Instrumente precum programul Visio automatizează procesul creării modelelor logice ale bazelor de date și creării bazelor de date fizice după modele. ERD-urile sunt alcătuite din entități, atributele entităților și relațiile dintre entități.

Dacă există un sfat cheie, acesta ar fi „Nu vă îngropați prea repede în detalii". Dacă vă ocupați de copaci, feriți-vă de pădure. Acest capitol nu a intrat prea mult în detalii privind câmpurile cu informații din fiecare entitate. De ce? Pentru că acest lucru nu este absolut necesar în realizarea unui proiect inițial solid al bazei de date. De fapt intrarea prea rapidă în detalii poate fi un obstacol. Se vor modifica entitățile pe măsura definirii câmpurilor din ele? Cel mai probabil, da. Dar acest lucru este în regulă; totul face parte dim procesul de design.

Vă amintiți mantra răbdării? Dar reclama lui Paul Masson: „Nu vom vinde vinul înainte de momentul potrivit" ?

În capitolul următor, veți trece cu proiectul prin etapa următoare a dezvoltării sale – procesul de normalizare. La sfârșitul capitolului 4, designul bazei de date Time Entry and Billing va fi complet.



4

PROIECTAREA UNEI BAZE DE DATE: INTRODUCERE ÎN NORMALIZARE

În acest capitol:

- Ce înseamnă normalizarea unei baze de date?
- Pașii necesari pentru normalizare
- Denormalizarea datelor – ce rost are să încălcăm regulile?
- Normalizarea aplicată – revedere a TEB
- Rezumat

În ultimul capitol am făcut o introducere în conceptele de bază ale proiectării bazelor de date. În acest capitol vom consolida aptitudinile dobândite în cel precedent. Dacă vă reamintiți una dintre regulile de bază ale cărții, este nevoie de mai multe iterații pentru a obține un design optim al bazei de date. După ce au fost create primele tabele, pasul următor presupune completarea acestora cu coloane. În acest moment, singurelele coloane care există sunt cheile primare și cele externe necesare pentru a suporta relațiile dintre tabele. În acest punct, întrebarea este „Ce coloane trebuie incluse în tabele?". Unele dintre coloanele pe care trebuie să le includeți sunt logice; altele nu sunt. Așa cum veți vedea, includerea unei coloane poate determina crearea unor noi tabele. Dacă vă gândiți că au fost create toate tabelele necesare pentru baza de date Time Entry and Billing (TEB), veți avea o surpriză! Procesul determinării coloanelor necesare într-un tabel se numește ***normalizare***. Procesul de normalizare este scopul acestui capitol, iar la sfârșitul său proiectarea bazei de date TEB va fi completă.

Ce înseamnă normalizarea unei baze de date?

Normalizarea unei baze de date poate fi caracterizată cel mai bine ca procesul organizării acesteia. Acum că știm ***ce*** înseamnă, să revenim la ***de ce***. Scopul normalizării este reducerea problemelor de consistență a datelor, prin reducerea redundanței. Pare derutant? Conseptul de normalizare poate fi înțeles mai ușor prin intermediul unui exemplu simplu. Pentru a aceasta, să revenim la baza de date Northwind Traders din Access.

Figura 4.1 prezintă ERD-ul (Entity Relationship Diagram) pentru baza de date Northwind Traders care a fost introdusă în capitolul 2. Observați relația dintre tabelele Customers și Orders. Ca recapitulare a funcționării relațiilor într-o bază de date, cheia primară a tabelului părinte este preluată în tabelul copil ca și cheie externă. În acest caz, câmpul CustomerID este preluat în tabelul Orders. Iată cum înregistrările de comenzi de la un anumit client pot fi asociate acestuia.

Atunci, de ce nu am prelua și alte câmpuri din tabelul Customers? Așa cum veți vedea mai târziu, uneori veți alege să preluați alte câmpuri din tabelul părinte în tabelul copil. Ca regulă generală, totuși, nu veți dori să faceți acest lucru. De exemplu, dacă veți decide să preluați câmpul CompanyName în tabelul Orders și se va modifica numele firmei? Va trebui să actualiza actualizați și tabelul Customers și tabelul Orders, precum și orice alt tabel care mai conține câmpul CompanyName. Continuând exemplul, ce s-ar întâmpla dacă diferitele apariții ale acestui câmp nu ar fi actualizate simultan? Care versiune va reflecta cel mai corect numele firmei? Dacă vi se va cere să pregătiți un raport care trebuie să conțină acest câmp, ce tabel veți folosi ca sursă?

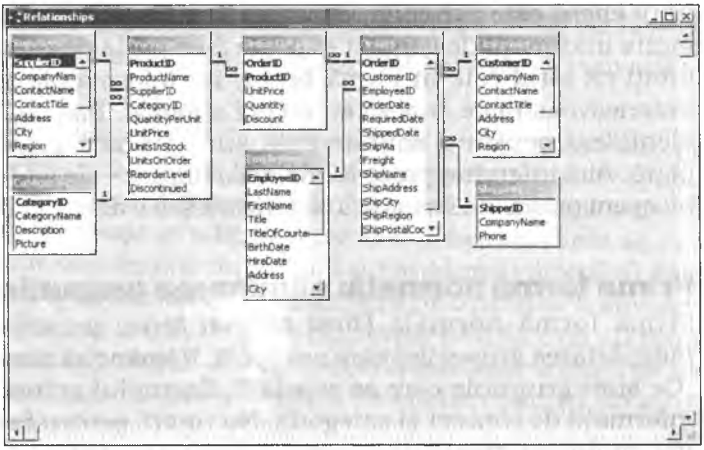


FIGURA 4.1
Diagrama de relații între entități pentru baza de date Northwind Traders prezintă relația dintre clienți și comenzi.

Este destul de clar că existând doar o apariție a elementului de date respectiv, nu mai rămâne nimic de interpretat și de ghicit. Când actualizați câmpul CompanyName în tabelul Customers, puteți fi sigur că toate rapoartele care se bazează pe acest câmp vor fi corecte și actuale. De ce? Pentru că respectivul câmp nu este preluat în mod redundant în baza de date, și, ca rezultat, este asigurată consistența datelor. Același concept poate fi revăzut și în modelul de date din figura 4.1. O consistență mai mare prin eliminarea redundanțelor – acesta este scopul normalizării!

Pașii de normalizare a modelului de date

Așa cum realizarea unui design optim al bazei de date este un proces cu mai mulți pași, la fel este și procesul de normalizare. Veți dori să normalizați până la a treia formă normală, cel puțin. Foarte posibil, veți dori să faceți încă un pas normalizând până la forma Boyce-Codd. Formele a patra și a cincea normale există, dar utilizarea lor este mult mai rară. Pentru scopuri informaționale, și acestea sunt descrise pe scurt la sfârșitul acestei secțiuni.

Pentru a ilustra procesul de normalizare, considerați următoarea listă de date nenormalizate:

- Identificatorul clientului (Customer ID) – cheie primară;
- Numele clientului (Customer Name);
- Tipul clientului (Customer Type);
- Nume de contact (Contact Name) – unu la mulți;
- Numele categoriei (Category Name) – unu la mulți.

Dacă ați lucrat vreodată cu date provenind dintr-o sursă nerelațională, acest scenariu vă poate fi familiar. În acest exemplu, lungimea înregistrării

unui client este variabilă, deoarece fiecare client poate avea una sau mai multe informații de contact și poate face parte din mai multe categorii. Nu uitați că totul este în formă brută, iar informațiile produse de eforturile dumneavoastră de analiză sunt nerelaționale. Toate atributele pe care le-ați identificat pentru o entitate dată sunt grupate probabil într-o structură plată. Aici intervine procesul de normalizare – de a organiza atributele. Să începem prin a lua structura anterioară și a o aduce la prima formă normală.

Prima formă normală: eliminarea grupurilor care se repetă

Prima formă normală (***first normal form***, pe scurt 1NF) presupune îndepărtarea grupurilor care se repetă. Rămâne să răspundem la întrebarea „Ce sunt grupurile care se repetă?". Exemplul anterior are două grupuri: informații de contact și categoria. Nu uitați, pentru fiecare client pot exista mai multe informații de contact și mai multe categorii.

Pentru fiecare grup ce se repetă pe care îl întâlnim, grupul este mutat într-un tabel separat. În acest caz, veți avea în final două tabele care conțin datele privind informațiile de contact și categoria. Schița următoare prezintă noua structură și entitățile sale:

- Tabelul Customer:
  - Identificatorul clientului;
  - Numele clientului;
  - Tipul clientului;
- Tabelul cu informații de contact:
  - Identificatorul contactului;
  - Identificatorul clientului;
  - Nume de contact.
- Tabelul privind categoria:
  - Identificatorul categoriei;
  - Identificatorul clientului;
  - Numele categoriei.

Așa cum poate ați ghicit, tabelul Customer este părinte pentru tabelele Contact și Category. Cele două relații sunt unu la mulți. Cu alte cuvinte, fiecare client poate avea unul sau mai multe contacte și poate fi asociat cu una sau mai multe categorii. Dar care este cheia primară pentru fiecare tabel? Gândiți-vă la asta câteva minute. Problema cheii primare este strâns legată de a doua formă normală, subiectul secțiunii următoare. În capitolele anterioare, cheia primară a fost întotdeauna formată dintr-un singur câmp. Deși acesta a fost cazul tabelului Customer, se poate spune același lucru și

despre tabelele Contact și Category? Răspunsul nu este atât de tranșant cum s-ar părea. Gândiți-vă în continuare la acest subiect – îl vom relua în secțiunea următoare.

Înainte de a trece mai departe, este important să vedeți beneficiile care derivă din mutarea informațiilor de contact și categoriei în tabele separate. Imaginați-vă cât de dificilă ar fi sarcina administrării informațiilor de contact dacă acestea ar fi stocate în tabelul Customer. Dacă un client ar putea avea un singur nume de contact, ar putea fi un argument faptul că datele precum nume, număr de telefon și așa mai departe ar putea fi stocate în tabelul Customer. Până la urmă, în acest caz, avem de a face cu un număr finit de coloane. Dar dacă cineva ar solicita o îmbunătățire a bazei de date care implică suportul mai multor informații de contact pentru un singur client? Aveți două variante:

- Adăugați alte coloane în tabelul Customer pentru a suporta contactele multiple;
- Adăugați un tabel copil care permite orice număr de contacte.

Evident, varianta a doua este mai ușoară, mai flexibilă și mai economicoasă din punctul de vedere al costurilor. Pentru varianta 1, designul bazei de date ar putea să se modifice în continuare ori de câte ori un anumit client va avea mai mule contacte decât coloanele din tabelul Customer. Dacă este vreun lucru care poate schimba destinul unei aplicații pentru baze de date, acesta este continua schimbare a structurii și modelului acesteia. Un model solid și stabil al bazei de date este fundamentul pe care se sprijină toate celelalte. După dezvoltarea unei aplicații, modificarea structurii bazei de date poate fi foarte scumpă și cronofagă. Întreaga aplicație trebuie să fie testată amănunțit pentru a elimina posibilitatea introducerii unor defecte ca rezultat al modificării structurii acesteia. Rezolvarea acestor aspecte încă de la început evită apariția multor probleme care ar putea apărea mai târziu.

Diversele forme normale sunt în cascadă. Cu alte cuvinte, înainte de a ataca a doua formă normală, va trebui să aplicați prima formă normală. Acest lucru este valabil și pentru următoarele forme normale. Și cum prima formă normală este realizată, să trecem la a doua formă normală, care presupune eliminarea datelor redundante. Cum vor fi afectate tabelele de acest lucru? Veți găsi răspunsul în secțiunea următoare.

A doua formă normală: eliminarea datelor redundante

Pentru a aduce tabelele la a doua formă normală, trebuie să analizați câmpurile în raport cu cheia primară. Problema cheilor primare a fost discutată în secțiunea anterioară. Cheia primară a tabelelor Contact și Category este multivalorică, adică nu aveți un singur câmp care să identifice în mod unic o înregistrare. Pentru tabelul Category, cheia primară este o combinație a câmpurilor Category ID și Customer ID. Mulți clienți pot utiliza aceeași categorie, iar un client poate face parte din mai multe categorii.

Totuși, un client nu poate face parte de mai multe ori din aceeași categorie. Acesta pare să fie un lucru logic. Datorită acestei reguli, combinația câmpurilor Category ID și Customer ID identifică în mod unic o înregistrare din tabelul Category. Să revenim la tabelul Contact. Așa cum veți vedea, acest scenariu prezintă o situație ambiguă.

Câmpul Contact ID identifică o înregistrare în mod unic? Ar putea să identifice unic înregistrarea; dar la fel de bine, ar putea să n-o facă. Ce părere aveți de acest răspuns? Răspunsul depinde de context. Tabelul Contact se referă numai la datele de contact? Sau mai corect ar fi să fie denumit Customer Contacts (date de contact despre clienți)? Din nou, răspunsul depinde de context. Poate fi un contact asociat mai multor clienți? Sau este asociat unui singur client? Analiza și eforturile de proiectare pe care le faceți ar putea da răspunsul la aceste întrebări. Uneori, corporațiile mari pot să apară de mai multe ori în tabelele cu clienți. Este foarte posibil ca o persoană de contact să fie asociată mai multor înregistrări ale clienților, de aceea este foarte important ca analiza și eforturile dumneavoastră de proiectare să descopere toate detaliile ca acesta. După implementarea unui sistem, corectarea unei greșeli poate fi foarte dificilă.

CÂT COSTĂ CORECTAREA UNEI GREȘELI?

Dacă este vorba despre o bază de date de 1 dolar, corectarea cât mai rapidă a unei greșeli poate costa 1 dolar. Cât mai rapid posibil înseamnă întimpul prcesului de analiză, design și modelare a bazei de date. După crearea bazei de date costul corectării aceeleiași greșeli este de 10 dolari. După ce sistemul a început faza de testare beta, greșeala costă 100 de dolari. Dacă greșeala trece neobservată la testarea beta și intră în producție repararea ei va costa 1000 de dolari.

Dacă vă gândiți mai bine, este logic să fie așa. După intrarea în faza de testare beta, dacă este descoperită o greșeală, trebuie să o luați de la început. Imaginați-vă că aplicația dumneavoastră este instalată în câteva sute de locații. Costul de a redistribui programul și de a actualiza baza de date poate fi imens. Ideea care se desprinde de aici este că, pe măsură ce treceți prin diferite stadii de dezvoltare, costul reparării defectelor se mărește de câte 10 ori. Cu cât analiza și proiectarea pe care le realizați sunt mai eficiente, cu atât este mai probabil să nu depășiți bugetul stabilit și să livrați aplicația la timp sau în avans.

Pentru scopul de a exemplifica, să presupunem că un contact poate fi asociat unui singur client. Gândindu-ne la acest lucru, tabelul Contact se află în a doua formă normală, deoarece atributele nu depind de o cheie multivalorică, adică numele de contact depinde numai de identificatorul contactului. În acest caz, câmpul Contact ID identifică în mod unic înregistrarea unui contact. De asemenea, nu există date redundante. Dar tabelul Category este în a doua formă normală? De data asta răspunsul este nu.

În timp ce un contact este specific unui client, o categorie este un obiect independent. Pentru a exemplifica să presupunem că avem categoriile medical, marketing și financiar. Un client poate aparține și categoriei marketing și categoriei financiar. Fiecare dintre aceste categorii poate fi asociată cu mai mulți clienți. Iată, de exemplu, datele următoare, care ar putea să se găsească în tabelul Category:

Category ID	Customer ID	Categoria
1	1	Medical
2	1	Marketing
3	2	Financiar
2	2	Marketing

Sunt două întrebări:

- Care este cheia primară?
- O categorie depinde integral sau parțial de cheia primară?

În acest caz, cheia primară este multivalorică, ea combinând câmpurile Category ID și Customer ID. Categoria depinde parțial, nu integral, de cheia primară. Dacă există un câmp care nu depinde integral de cheia primară, atunci tabelul nu este în a doua formă normală. Pentru a aduce acest tabel în a doua formă normală, categoriile trebuie plasate într-un tabel separat. Aceasta mărește numărul de tabele de la trei la patru. În continuare prezentăm noua structură a tabelului Category:

- Tabelul Category:
  - Category ID (identificatorul categoriei);
  - Category Name (numele categoriei).

Mai jos este prezentată structura noului tabel Customer Category:

- Tabelul Customer Category:
  - Customer Category ID (identificatorul categoriei clientului) – cheie primară;
  - Customer ID (identificatorul clientului);
  - Category ID (identificatorul categoriei).

CHEI PRIMARE ȘI CHEI PRIMARE SUROGAT

Aveți nevoie de un câmp Customer Category ID în tabelul Customer Category? Răspunsul este nu. Combinația câmpurilor Customer ID și Category ID va servi ca și cheie primară multivalorică. Acest tip de cheie primară mai este numit și cheie primară ***compusă***. Într-un tabel precum Customer Category, care are numai rolul de tabel de rezolvare a relației mulți la mulți, o cheie compusă este o alternativă perfect acceptabilă. Este foarte puțin probabil ca tabelul Customer Category să se

comporte ca părinte, deci cheia primară probabil nu va fi preluată într-un tabel copil. Dacă alegeți să păstrați o cheie primară univalorică sau multivalorică este o chestiune de preferință personală.

Includerea câmpului este o problemă de consistență și preferință pentru o anumită metodologie. Cele două metodologii în chestiune implică utilizarea sau neutilizarea cheilor surogat.

Există două curente de idei privind cheile primare. Unul dintre ele susține că acestea ar trebui produse în ordine secvențială; altul susține că ele ar trebui să fie combinații ale câmpurilor existente. Cheile surogat sunt produse în ordine secvențială, folosind un dispozitiv de incrementare. În Access este utilizat în acest scop tipul de câmp autonumeric (autonumber). Uneori, cheile surogat sunt numite chei fără semnificație, deoarece din valoarea acestora nu sunt obținute informații privind înregistrările pe care le identifică. Celălalt tip de chei, cheile multivalorice, este alcătuit dintr-o combinație a câmpurilor existente. Uneori, cheia este o combinație între numele clientului+adresa+oraș+stat. Este valabilă orice informație care identifică în mod unic o înregistrare. Este ușor de văzut că aceste chei multivalorice nu au semnificație. Când vedeți valoarea unei chei, vedeți imediat datele descrise de înregistrarea pe care cheia o identifică.

Care metodă este corectă? Ambele au calitățile lor. Dintre cele două, cheile surogat reprezintă probabil o abordare mai bună. Pentru a analiza de ce sunt o abordare mai bună, va trebui să înțelegeți scopul unei chei primare. O cheie primară servește la identificarea unică a unei înregistrări. Cheile primare sunt preluate în datele copil cu scopul de a lega înregistrările copil de cele părinte. Ce este mai ușor, preluarea unei singure valori de tip single integer sau a unui șir care poate conține peste 100 de caractere? Ce se va întâmpla dacă se modifică numele firmei, adresa sau orice altă componentă a cheii primare? Nu se va schimba numai cheia primară, ci și cea preluată ca și cheie externă în tabelele copil trebuie să se schimbe. Problemele de întreținere pe care le ridică cheile multivalorice depășesc orice avantaj pe care acestea l-ar putea avea.

Scopul unei chei primare este dublu: de a identifica în mod unic o înregistrare și de a servi drept „adeziv" între tabelele înrudite. Valoarea sa ar trebui să nu se schimbe. Cheile primare stau în fundal și mențin ordinea. În anumite privințe, ele sunt similare arbitrilor de fotbal. În meciurile frumoase, nu observați arbitrul. Utilizatorii primari nu ar trebui să vadă, să cunoască sau să controleze înfățișarea cheilor primare. Cunoscând toate acestea, sunt cheile primare singura cale de a identifica unic înregistrările? Răspunsul este negativ. Să luăm în considerare câmpul cu numărul clientului.

Doi clienți nu pot avea același număr. Altfel, de ce ar mai exista numărul clientului. Adesea, companiile au o metodă sistematică de a numerota clienții. O parte a numărului poate da informații despre client, cum ar fi numele său. În acest caz numărul clientului nu ar trebui să fie cheie primară, deoarece depinde de caracteristicile din înregistrarea cientului. Nu uitați, dacă se modifică un aspect de

care depinde numărul clientului, trebuie modificat și acest număr. Desigur, nu există nici un motiv pentru care cheia primară nu ar putea avea și un al treilea rol, cum ar fi cel de număr al clientului. Dacă așa stau lucrurile, utilizatorii nu ar trebui să aibă controlul asupra valorii efective a acestuia.

În continuare vom prezenta modul cum apar niște date în tabelul Category:

Category ID	CategoryName
1	Medical
2	Marketing
3	Financiar

Iată cum ar putea să apară niște date în tabelul Customer Category:

CustomerCategory ID	Customer ID	Category ID
1	1	1
2	1	2
3	2	1
4	2	2

Înainte de a aduce tabelele la a doua formă normală, considerați scenariul modificării descrierii unei categorii. În cazul datelor redundante vor fi necesare multe actualizări. După aducerea la a doua formă normală, descrierea unei categorii există într-o singură locație. După modificarea unei singure date, toate rapoartele care o conțin vor fi actualizate automat. Cât despre rapoarte, să presupunem că înainte de aducerea la a doua formă normală trebuie să creați un raport care afișează câmpul Category Name. Ce veți utiliza ca sursă pentru acest câmp? După aducerea la a doua formă normală, răspunsul este mult mai clar.

Să aruncăm o privire asupra tabelului Customer. Dacă examinăm câmpul Customer Type, se pare că mai există un grup care se repetă. Iată cum apar datele în acest tabel:

Cusomer ID	Customer Name	Customer Type
1	Ace Tomato	Business
2	PMH	Hospital
3	Rutgers Law	School
4	Microsoft	Business

Pentru a aplica a doua formă normală, trebuie creat un nou tabel Customer Type. Iată care va fi structura acestui tabel.

Customer Type ID	Customer Type Description
1	Business
2	Hospital
3	School

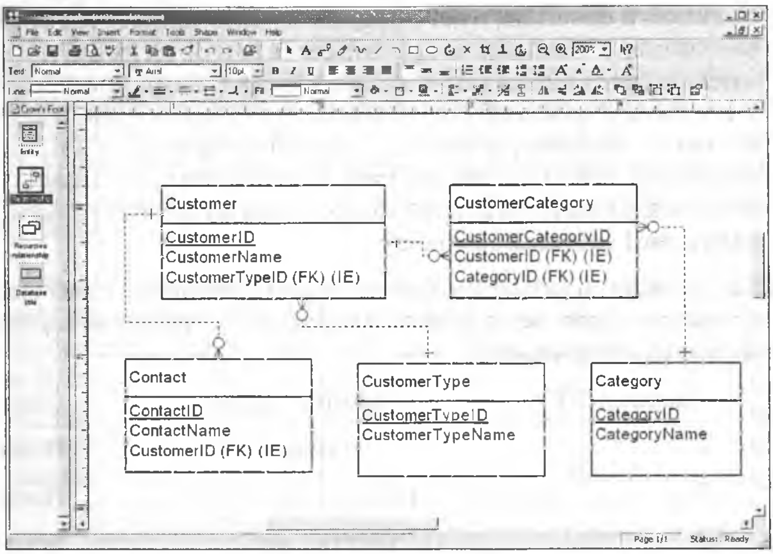
După crearea noului tabel Customer Type, trebuie modificat tabelul Customer, pentru a putea fi stabilită o legătură între cele două tabele. Iată care va fi structura noului tabel Customer:

Customer ID	Customer Name	Customer type
1	Ace Tomato	1
2	PMH	2
3	Rutgers Law	3
4	Microsoft	1

Figura 4.2 prezintă noul model de date produs prin normalizare.

FIGURA 4.2

Tabelele din acest model de date sunt aduse la a doua formă normală.



A treia formă normală: eliminarea coloanelor care nu depind de chei

Ați început cu o singură structură plată. Pentru a trece la prima formă normală, grupurile care se repetau au fost mutate în tabele separate. Au rezultat trei noi tabele: Contact, Category și Customer Type. Pasul următor este de a duce tabelele la a treia formă normală. Pentru a ilustra acest proces, trebuie să fie adăugate în tabelul Customer unele dintre câmpurile nou create:

- Customer ID;
- Customer Name;
- Customer Type ID;
- City (orașul);
- State (statul);
- ZIP Code (codul poștal).

Tabelul Customer este în prima formă normală deoarece nu există grupuri care se repetă și este în a doua formă normală deoarece nu există chei multivalorice. În acest caz, există o singură cheie primară surogat. Se pune problema dacă tabelul este în a treia formă normală. Pentru aceasta, ar trebui ca orice câmp care depinde de orice altceva în afară de cheia primară să fie mutat în alt tabel. Tabelul în care este mutat câmpul poate fi unul existent sau unul nou. Câmpurile precum orașul, statul și codul poștal depind de cheia primară? Răspunsul este nu. Aceste elemente sunt complet independente de cheia primară Customer ID.

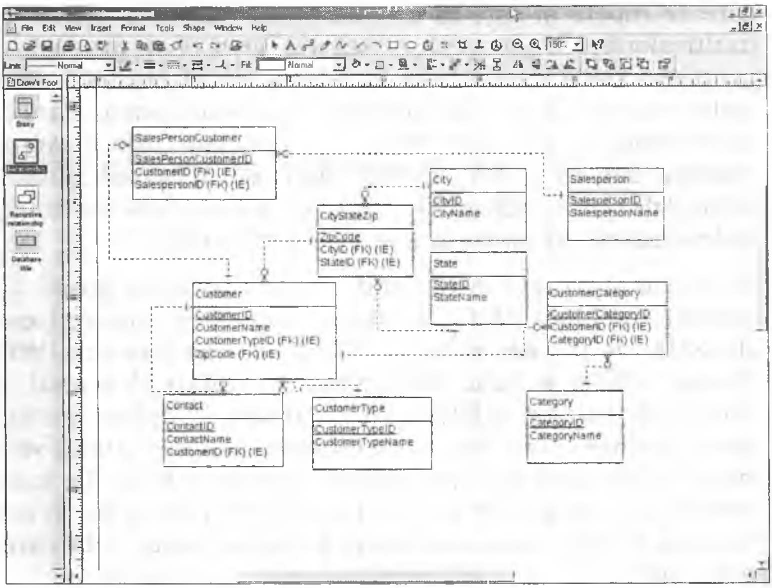
Să examinăm rapid relația dintre oraș, stat și cod poștal. Codul poștal este specific unei localități și unui stat (este vorba despre localitățile și statele din SUA – n. tr.). de exemplu, 19301 este codul pentru Paoli, Pennsylvania. Există, totuși, și Paoli, Oklahoma. Nu uitați că scopul normalizării este reducerea datelor redundante. Va trebui să definiți numai un singur element de date și să-l reutilizați după necesități. Gândiți-vă că un mare oraș precum Philadelphia are sute de coduri poștale. Pe măsură ce studiați problema, devine clar că este necesar un nou tabel. În acest caz veți crea un nou tabel pentru stat, oraș și codul poștal. Iată care va fi structura acestuia:

- Tabelul State:
  - State ID (identificatorul statului);
  - State Name (numele statului).

- Tabelul City:
  - City ID (identificatorul orașului);
  - City Name (numele orașului).
- Tabelul City State ZIP:
  - ZIP Code (codul poștal);
  - City ID (identificatorul orașului);
  - State ID (identificatorul statului).

Cum sunteți sigur de unicitatea codului poștal, îl puteți utiliza ca și cheie primară. Figura 4.3 arată cum apare modelul de date după aplicarea celei de a treia forme normale tabelului Customer. Așa cum veți vedea când va fi discutat subiectul denormalizării, aducerea tabelului Customer la a treia formă normală s-ar putea să nu aducă rezultate optime.

FIGURA 4.3
Tabelele din acest model de date sunt aduse la a treia formă normală.



În general, nu veți trece mai departe de a treia formă normală. Ar trebui să nu uitați însă că există și a patra și a cincea formă normală. Utilizarea lor este rară, totuși. În scopul completitudinii secțiunea următoare prezintă pe scurt și aceste forme normale.

A patra și a cincea formă normală

În majoritatea cazurilor, nu veți trece mai departe de a treia formă normală. Asta nu înseamnă că povestea normalizării s-a terminat. Există și a patra și a cincea formă normală. A patra formă normală izolează relațiile independente multiple. Dar ce înseamnă asta? Sunt o mulțime de cuvinte, dar conceptul nu este greu de înțeles. Să începem cu a patra formă normală.

Pe măsură ce efectuați analiza, determinați categoriile de clienți care se încadrează în anumite intervale ale volumului de vânzări. Știind ceea ce știți despre normalizare, decideți să creați un tabel de definire a intervalelor cu volume de vânzări și apoi să legați acest tabel de tabelul Customer Category. Iată în continuare structura revizuită a tabelelor afectate de această modificare:

- Tabelul Customer Category:
  - Customer Category ID – cheie primară;
  - Customer ID;
  - Category ID;
  - Sales Range ID (indentificatorul intervalului de vânzări).
- Tabelul Sales Range (intervalele vânzărilor):
  - Sales Range ID (identificatorul intervalului de vânzări) – cheie primară;
  - Sales Range Description (descrierea intervalului de vânzări).

Se pune întrebarea dacă există relații reale între tabelele Customer Category și Sales Range. Cel mai probabil, răspunsul este nu. Clientul are vânzări, dar, intervalele în care se încadrează vânzările sunt asociate unui client ca niște categorii. Pentru aducerea la a patra formă normală, câmpul Sales Range ID trebuie să fie mutat în tabelul Customer. În cazul ipotetic curent, tabelele Customer, Contact, Category și Customer Category satisfac condițiile celei de a patra forme normale.

Conceptul din a cincea formă normală este ceva mai greu de înțeles. Aceasta presupune divizarea tabelelor aduse la a patra formă normală în scopul reducerii numărului de rânduri care trebuie introduse, modificate sau șterse în timpul diverselor operații de actualizare. Se pune întrebarea dacă este mai ușoară întreținerea câtorva rânduri de date în plus sau întreținerea unui tabel suplimentar. Dacă chiar nu aveți nevoie de un nou tabel, mai bine evitați crearea lui. Pentru a ilustra aducerea la a cincea formă normală, vom considera exemplul următor.

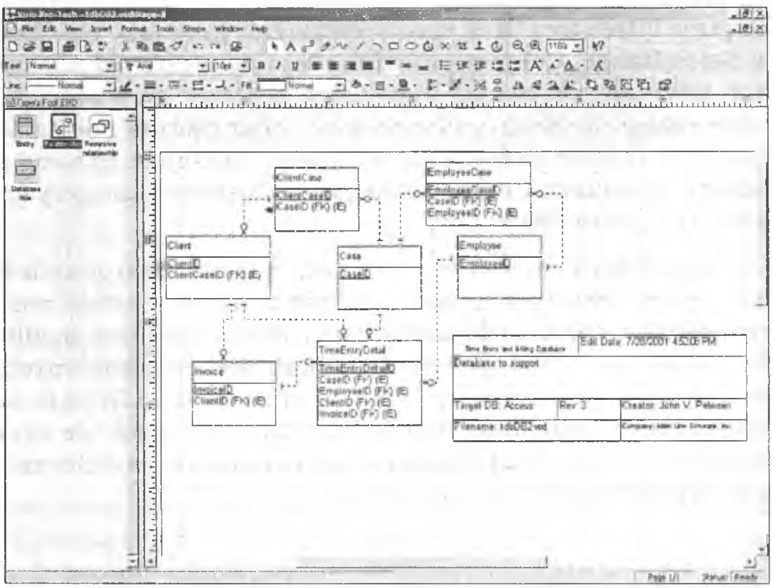
Să adăugăm un tabel Salesperson (agenți de vânzări) în baza de date ipotetică la care lucrați. Baza de date poate înregistra categoriile asociate unui client. În plus ea înregistrează și agentul de vânzări căruia i-a fost atribuită o anumită combinație de categorii de clienți. Urmarea este că trebuie să exisiste un tabel categorii-de-clienți-agent-de-vânzări pentru a susține asocierea între cele trei entități. Întrucât există o relație reală între categorii și agenții de vânzări, acest nou tabel se află în a patra formă normală.

Acum, să presupunem că este pusă în aplicare o nouă regulă: după ce unui agent de vânzări îi este asociată o combinație de categorii de clienți, acesta are acces la alte categorii cu care clientul ar putea fi asociat. Dacă un client este asociat cu trei agenți de vânzări, de fiecare dată când l-ați asociat cu o nouă categorie, trebuie să adăugați trei înregistrări în tabelul categorii-de-clienți-agent-de-vânzări – câte una pentru fiecare agent. Aceeași situație ar apărea în cazul în care clientul ar fi asociat cu trei categorii și ar interveni o modificare în personalul de vânzări. În acest caz, va trebui să actualizați trei înregistrări. A cincea formă normală presupune divizarea tabelului pentru a reduce numărul de înregistrări.

Pentru a îndeplini acest scop, vor fi create tabelele Customer Salesperson și Customer Category. Această explicație presupune că tabelul Customer Category nu exista. Analiza ne conduce, mai degrabă, direct la un tabel Customer Category Salesperson. Figura 4.4 prezintă modelul de date și noile entități care se află în a cincea formă normală.

FIGURA 4.4

Tabelele din acest model de date sunt în a cincea formă normală.



Normalizarea: câteva idei finale

Pe măsură ce câștigați experiență în dezvoltarea bazelor de date, veți descoperi că normalizarea devine un proces mai mult implicit decât unul explicit. Cu alte cuvinte, în timp ce organizați și modelați informațiile dobândite anterior, veți crea imediat tabele care se află în a treia formă normală câtă vreme folosiți chei surogat univalorice. Dacă aveți o relație mulți la mulți, așa cum este cazul între clienți și categorii va trebui să faceți un pas în plus și să treceți la a patra formă normală. Cât timp tabelul care joacă rol de joncțiune – tabelul Customer Category în acest caz –conține numai minimul de date necesare pentru a susține relația mulți la mulți, nu va trebui să vă faceți griji pentru trecerea la a patra formă normală, deoarece tabelul respectă deja condițiile necesare pentru această formă. În final, dacă nu trebuie să vă faceți griji pentru a patra formă normală, nu va trebui să vă faceți griji nici pentru a cincea formă normală. Din nou, tabelul va respecta și condițiile necesare pentru această formă. S-ar mai putea spune și că nu trebuie să vă faceți griji pentru a patra și a cincea formă normală, deoarece ele nu sunt aplicate în general. Câtă vreme ați ajuns la a treia formă normală, iar analiza și designul bazei de date au fost realizate cu minuțiozitate, șansele dumneavoastră de succes sunt foarte mari!

Denormalizarea datelor – ce rost are să încălcăm regulile?

Să luăm în considerare tabelele pentru oraș, stat și codul-poștal-oraș-stat. Pe de o parte, aceste tabele se află în a treia formă normală. Pe de altă parte, de fiecare dată când trebuie să scrieți o interogare pentru a extrage date despre clienți, va trebui să reuniți date din patru tabele. Ori de câte ori normalizați datele, va trebui să puneți în balanță două elemente contradictorii: scopul de a reduce redundanța datelor și munca suplimentară pe care trebuie să o desfășurați când va trebui să extrageți informații din baza de date. În majoritatea cazurilor, elemente precum orașul, statul și codul poștal sunt preluate redundant în tabele. Gândiți-vă la situația când încercați să atribuiți un oraș unui client, dar orașul nu a fost definit în tabelul cu orașe. Va trebui să actualizați mai întâi acest tabel. Acum, ceea ce s-ar fi realizat printr-o singură operație, necesită două operații. Asta nu înseamnă că nu ar exista un motiv întemeiat din punctul de vedere al firmei de a separa informațiile despre oraș, stat și cod poștal în mai multe tabele. În acest caz, are rost să încălcați regulile și să denormalizați baza de date.

Dar este important să urmați regulile pentru denormalizare. În același timp, este important să recunoașteți situațiile în care este necesar să încălcați regulile. Dacă nu le respectați asigurați-vă că aveți un motiv serios pentru a face acest lucru. Reducerea complexității interogărilor și îmbunătățirea performanțelor acestora sunt motive întemeiate pentru a face acest lucru. Fiindcă încălcarea regulilor trebuie să fie o excepție și nu situația normală. De aceea dacă observați că denormalizați în continuare, faceți o pauză pentru

a vă gândi la ceea ce faceți. Există posibilitatea să aplicați în mod eronat conceptul denormalizării.

Aplicarea normalizării – recapitulare a bazei de date TEB și rafinarea designului său

După ce ați înțeles bine normalizarea, este momentul să aplicați conceptele sale în proiectul curent, baza de date Time Entry and Billing (notată TEB). Pentru a recapitula, au fost deja identificate următoarele tabele:

- Case (cazuri);
- Client (clienți);
- ClientCase (cazuri-clienți);
- Employee (angajați);
- Invoice (facturi);
- TimeEntryDetail (detalii privind timpul alocat).

Figura 4.5 prezintă relațiile dintre entități.

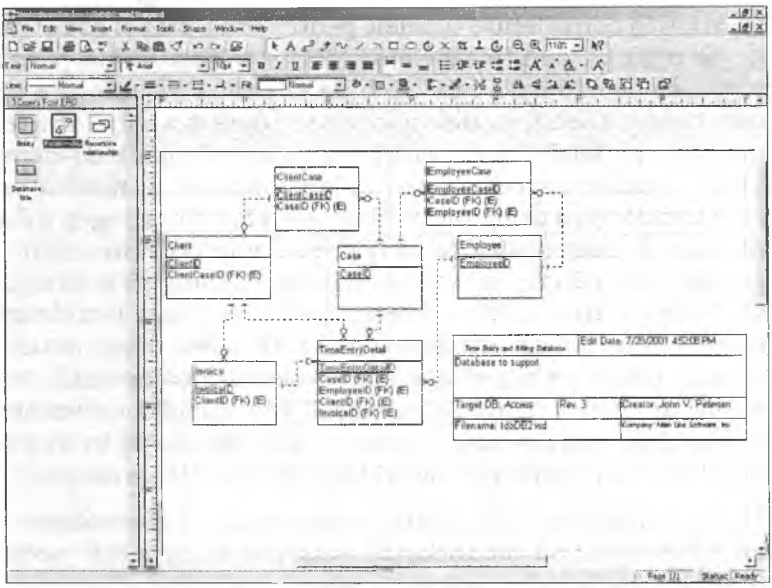


FIGURA 4.5
Versiunea 4 a modelului bazei de date TEB reprezintă prima schiță completă a modelului bazei de date.

Identificarea noilor coloane și tabele

Așa cum bănuiți, probabil, vor fi necesare noi tabele. Înainte de a face acest pas, tabelele existente trebuie completate cu coloane. Dar până să trecem mai departe, este important să reținem că nu au fost examinate toate aspectele privind datele de pontaj. Modelul cu care lucrați conține informațiile de bază necesare pentru a exemplifica și a preda conceptele despre baze de date tratate în această carte. Să construim acum modelul de date început în capitolul 3.

Tabelele EmployeeCase și ClientCase

Dacă vă mai amintiți, EmployeeCase și ClientCase sunt tabele de joncțiune pentru relații mulți la mulți. Unui client i se pot atribui mai mulți angajați. În plus, unui angajat i se pot repartiza mai multe cazuri. Același tip de relație există între clienți și cazuri. Datorită naturii acestor tabele, ele nu necesită coloane suplimentare.

Tabelul Employee

Să începem cu tabelul Employee. Ce fel de informații trebuie stocate în el? Sunt necesare cel puțin următoarele date:

- Employee ID (identificatorul angajatului);
- First Name (prenumele);
- Middle initial (inițiala celui de-al doilea nume);
- Last name (numele de familie);
- Social Security Number (numărul asigurării sociale);
- Address 1 (adresa 1);
- Address 2 (adresa 2);
- City, State, ZIP Code (oraș, stat, cod poștal);
- Home Phone and Work Extension (telefonul de acasă și interiorul de la serviciu).

În plus față de cele din lista de mai sus, mai sunt necesare câteva informații. Una dintre ele este clasificarea angajatului. În firma de avocatură există parteneri, asociați, juriști, secretare și personal administrativ. Oricând vă confruntați cu un element de clasificare, puteți paria că va fi necesar un nou tabel. Vă amintiți discuția despre a treia formă normală? Clasificarea unui angajat poate fi un element generic care nu depinde de cheia EmployeeID. Dacă vă gândiți la acest lucru, trebuie să adăugați un nou tabel EmployeeClassification, împreună cu cheia externă EmployeeClassificationID în tabelul Employee.

Firmele de avocatură au de obicei multe departamente. De exemplu, ar putea avea unul sau toate departamentele următoare: impozite, antitrust, litigii civile, proprietate intelectuală, asigurări și municipalitate, pentru a numi numai câteva. Pare a fi un nou element de clasificare, nu? Veți adăuga un nou tabel în baza de date, numit Department. În plus, va trebui să adăugați în tabelul Employee un câmp numit DepartmentID.

Sfat

Să ne amintim că tabelul Employee are următoarea structură:

- Employee ID (identificatorul angajatului) – cheie primară;
- Department ID (identificatorul departamentului) – cheie externă de la tabelul Department;
- Employee Class ID (identificatorul clasei angajatului) – cheie externă de la tabelul Employee Class;
- First Name (prenumele);
- Middle Initial (inițiala celui de-al doilea nume);
- Last Name (numele de familie);
- Social Security Number (numărul asigurării sociale);
- Address 1 (adresa 1);
- Address 2 (adresa 2);
- City, State, ZIP Code (orașul, statul, codul poștal);
- Home Phone and Work Extension (telefonul de acasă și interiorul de la serviciu);
- E-mail Address (adresa de e-mail).

În conjuncție cu noile chei externe, la modelul de date au fost adăugate următoarele tabele:

- Department;
- EmployeeClass;

În afară de câmpul DepartmentID, tabelul Department mai conține și un câmp numit Description (descriere).

Tabelul Cilent

Tabelul Client este destul de simplu. În el trebuie stocate următoarele elemente de date:

- ClientID (identificatorul clientului) – cheie primară;
- Client Type (tipul clientului) – firmă sau persoană fizică;
- First Name (prenumele);
- Middle Initial (inițiala celui de-al doilea nume);

- Last Name (numele de familie);
- Organization Name (numele organizației);
- Address 1 (adresa 1);
- Address 2 (adresa 2);
- City, State, ZIP Code (orașul, statul, codul poștal);
- Phone (telefonul);
- E-mail Address (adresa de e-mail).

Trebuie discutate câteva elemente. Mai întâi, pentru tipului clientului este necesar un alt tabel pentru a stoca diferitele tipuri de clienți pe care firma îi reprezintă? În acest caz, răspunsul este negativ. În timp ce vă desfășurați analiza, ați aflat că există numai două tipuri de clienți: firme și persoane fizice. Atunci fiecare înregistrare a unui client va avea un câmp numit Client Type (tipul clientului) care va conține fie litera B de la „business", fie litera I de la „individual". Într-un astfel de caz, nu există un beneficiu practic din crearea unui alt tabel.

Dar contactele? Nu este nici o îndoială că în cazul clienților care sunt firme va trebui să oferiți contacte. Întrebarea următoare este dacă pot exista contacte multiple pentru fiecare client. Cel mai probabil, răspunsul este da. Veți avea nevoie de informații de contact despre mai mulți dintre angajații firmei client. În concluzie, dacă vă gândiți că va trebui să definiți un tabel denumit Contact (informații de contact), sunteți pe calea cea bună!

Indiciu

Structura tabelului Contact este simplă:

- ContactID (identificatorul contactului) – cheie primară;
- First Name (prenumele);
- Middle Initial (inițiala celui de-al doilea nume);
- Last Name (numele de familie);
- Title (titlul);
- Phone (telefonul);
- Extension (interiorul);
- E-mail Address (adresa de e-mail).

Ce legătură are câmpul Title cu necesitatea unui nou tabel pentru căutare? Titlurile posturilor sunt elemente generice. În același timp, ele pot varia de la firmă la firmă. Iată de ce trebuie să decideți dacă veți stoca informațiile într-un tabel sau le veți lega datele cu un alt tabel și o cheie externă. Avantajul de a avea titluri predefinite stocate într-un tabel este consistența datelor. De exemplu, ați putea controla variațiile de tipul CEO sau C.E.O. în același timp, probabil că vor exista numai câteva titluri. Când vă îndreptați atenția spre construirea interfeței cu utilizatorul, vi se vor prezenta

niște tehnici pentru respectarea standardelor care vor permite variațiile. În concluzie puteți să creați sau să nu creați un tabel separat cu titlul postului. Dacă nu veți crea un tabel separat, titlul va fi stocat în înregistrarea contactului.

Tabelul Case

Tabelul Case stochează informațiile de bază despre un caz, inclusiv titlul, judecătorul, tribunalul, departamentul, data începerii, data procesului, data înțelegerii și observații. Este necesar un tabel separat pentru judecător și tribunal? Cel mai probabil, răspunsul este da. Firma dorește să știe ce cazuri au mai fost prezidate de un anumit judecător. În plus, firma va dori să știe detalii privind cazurile care au fost judecate în anumite tribunale. Dacă luăm ca exemplu orașul Philadelphia, acolo există tribunal de stat, municipal și federal. În această clasificare există multe tipuri de tribunale în care poate fi judecat un caz. Bineînțeles, firma dorește să știe ce cazuri sunt judecate în ce tribunale.

Notă

Cazurile judecate de tribunale au un anumit calendar. În realitate, vom asocia tabelului Case un tabel copil pentru a ține evidența diferitelor elemente care depind de timp, cum sunt depozițiile, moțiunile de dinainte de proces, răspunsurile, audierile în fața judecătorului, discuțiile pentru înțelegeri și așa mai departe. Tabelul copil se va comporta ca un memento, pentru a anunța utilizatorii despre apropierea unor termene sau prezentări în fața instanței. Deoarece aceste elemente nu sunt direct înrudite cu contorizarea timpului și cu factura, ele vor fi ignorate.

Tabelul Case are următoarea structură:

- CaseID (identificatorul cazului) – cheie primară;
- DepartmentID (identificatorul departamentului) – cheie străină de la tabelul Department;
- CourtID (identificatorul tribunalului) – cheie străină de la tabelul Court;
- JudgeID (identificatorul judecătorului) – cheie străină de la tabelul Judge;
- Docketnumber (numărul procesului pe rol);
- Title (titlul);
- Notes (observații);
- Start Date (data începerii);
- Trial Date (data procesului);
- Settlement Date (data înțelegerii).

Fiecare caz trebuie să fie atribuit unui departament. De exemplu, un litigiu privind un patent ar trebui să fie atribuit departamentului Proprietate

Intelectuală. În ce fel ajută acest lucru? Când vine momentul de a atribui angajaților un caz, nu veți dori să faceți numai acest lucru. Mai curând veți dori să atribuiți cazul numai unor angajați care fac parte din departamentul respectiv. Așa cum știți, un angajat trebuie să aparțină unui departament. Așa cum veți vedea mai târziu, organizarea datelor în acest mod vă permite să filtrați lista angajaților. Astfel sunt evitate greșelile, ca de exemplu atribuirea unui litigiu privind proprietatea intelectuală unui avocat specializat în impozite. Mai târziu în această carte vom face o introducere în regulile dintr-o firmă și integritatea datelor. Așa cum veți vedea și cum probabil ați văzut deja, un design solid al bazei de date îmbunătățește capacitatea de a întări regulile firmei și integritatea datelor.

Noul tabel Court conține un câmp numit CourtID (identificatorul tribunalului) și unul numit Description (descriere). Tabelul Judge conține un câmp numit JudgeID (identificatorul judecătorului), precum și un câmp numit CourtID (identificatorul tribunalului) pentru a realiza legătura cu tabelul Court. În plus, sunt stocate: prenumele, inițiala celui de-al doilea nume, numele de familie și observațiile despre judecător. Chiar dacă nu vă dați seama, apare astfel o problemă interesantă. Tabelul Case are legături și cu tabelul Court și cu tabelul Judge. Tabelul Judge conține o legătură cu tabelul Court, deci judecătorului i se atribuie un tribunal. Dacă judecătorul are deja o legătură cu tribunalul, de ce este nevoie de câte o legătură la tabelul Case și de la tabelul Court și de la tabelul Judge? Pentru că este foarte posibil ca altui judecător să-i fie repartizat cazul dumneavoastră. De asemenea, aceste legături multiple sunt utile în realizarea rapoartelor.

Tabelul TimeEntryDetail

Să revenim acum la tabelul TimeEntryDetail. În timpul analizei, ați determinat deja că o înregistrare din acest tabel va face legătura cu o înregistrare din tabelul Invoice la momentul facturării. Fiecare înregistrare din tabelul TimeEntryDetail trebuie să aibă o legătură cu o combinație angajat-caz-client. Nu uitați că un caz poate implica mai mulți clienți. De aceea, nu este suficient să realizați doar o legătură la un caz. Problema alegerii clienților cărora li se vor întocmi facturi va fi foarte dificilă. Ce date mai trebuie stocate în tabelul TimeEntryDetail? Dacă ați completat vreodată o foaie de pontaj, probabil ghiciți care sunt cele mai importante unformații. Elemente precum activitatea, data, numărul de ore și tariful orar sunt un bun punct de pornire. În acest context, tariful orar este cel din factura întocmită clientului, nu cel plătit angajaților. Plata angajaților depășește scopul acestei baze de date. Mai există câteva probleme privind tariful orar, deci să ne oprim puțin pentru a le discuta.

Cum va plăti un client pentru serviciile care i se oferă? Depinde de natura activităților. De exemplu, pentru activități administrative, cum ar fi fotocopierea documentelor, clientul ar putea plăti 50 $ pe oră. Dacă la un caz lucrează un jurist, atunci clientul ar putea fi taxat cu 100 $ pe oră, iar dacă un avocat reprezintă clientul în instanță, clientul ar putea fi taxat cu

1500 $ pe oră. În prezent, informațiile privind prețurile obișnuite pentru diversele activități nu sunt stocate în baza de date. În tabelul TimeEntryDetail este stocat un tarif de facturare. Pentru a asigura consistența și acuratețea datelor, rata obișnuită trebuie să fie stocată undeva. Întrebarea este unde. Ce ați zice de tabelul Employee? Un angajat poate avea mai multe tipuri de activități? Răspunsul este da. Un avocat pledant poate discuta la telefon cu clientul, poate apărea în instanță sau poate lua o depoziție. Fiecare dintre aceste activități are diferite tarife orare. Pe baza acestei cerințe, tariful obișnuit depinde de activitatea desfășurată. Și atunci, categoria de activitate este stocată în prezent în baza de date? Răspunsul este negativ. De aceea, este necesar un nou tabel, numit WorkCategory (categoria activității). Detaliile din noul tabel WorkCategory vor fi discutate pe scurt, dar, înainte de a trece mai departe, vedeți cum procesul de analiză a unui tabel poate duce la crearea de noi câmpuri și tabele?

Tabelul TimeEntryDetail are structura următoare:

- TimeEntryDetailID (identificatorul detaliilor de pontaj) – cheie primară;
- InvoiceID (identificatorul facturii) – cheie străină de la tabelul Invoice;
- CaseID (identificatorul cazului) – cheie străină de la tabelul Case;
- EmployeeID (identificatorul angajatului) – cheie străină de la tabelul Employee;
- ClientID (identificatorul clientului) – cheie străină de la tabelul Client;
- Description (descriere);
- Hours (numărul de ore);
- Rate (tariful orar);
- Work Date (data desfășurării activității);
- Work Category (categoria activității).

În acest moment se pune întrebarea dacă înregistrările din tabelul TimeEntryDetail vor indica spre înregistrări înrudite din tabelul WorkCategory sau dacă informațiile vor fi preluate în înregistrările din tabelul TimeEntryDetail. În acest caz, trebuie ca înregistrările din tabelul TimeEntryDetail să fie independente de modificările din tabelul WorkCategory. De exemplu, ce s-ar întâmpla dacă, o anumită activitate este facturată cu 100 $ pe oră, iar peste trei săptămâni prețul aceluiași serviciu crește la 120 $ pe oră? Dacă există o legătură, la vizualizarea detaliilor înregistrării din tabelul TimeEntryDetail veți vedea noul tarif. Acest scenariu nu este funcțional. Mai mult tariful din tabelul WorkCategory este unul obișnuit. Tariful poate fi modificat după introducerea în înregistrarea din tabelul TimeEntryDetail. Din acest motiv, tariful actual trebuie stocat în înregistrările tabelului TimeEntryDetail. Întrucât descrierea categoriei activității se poate modifica, și aceasta va fi preluată în înregistrările tabelului TimeEntryDetail.

Cum veți ști că a fost facturată o anumită înregistrare din tabelul TimeEntryDetail? Poate vă gândiți că este necesar un câmp pentru a indica acest lucru. Aici intervine câmpul InvoiceID. El are rol dublu. Mai întâi, la

crearea unei noi înregistrări în tabelul TimeEntryDetail, acest câmp are valoarea implicită 0. Apoi, după facturare, noul câmp InvoiceID este completat cu o valoare. De aceea, valoarea 0 din câmpul InvoiceID în tabelul TimeEntryDetail arată că nu a fost facturat serviciul respectiv. Astfel veți ști dacă a fost sau nu întocmită factura.

Noul tabel WorkCategory are structura următoare:

- WorkCategoryID (identificatorul categoriei activității) – cheie primară;
- Description (descriere);
- DefaultRate (tariful obișnuit).

Dacă priviți ultima versiune a modelului bazei de date din figura 4.6, nu veți vedea o relație care să implice tabelul WorkCategory. În acest scenariu, tabelul WorkCategory are rol de șablon (***template***). Cu alte cuvinte, după utilizarea datelor dintr-o înregistrare a tabelului WorkCategory pentru completarea datelor în tabelul TimeEntryDetail, nu mai este necesară o referință la tabelul WorkCategory. Necesitatea creării și întăririi relațiilor este foarte importantă. Aceste concepte sunt discutate în capitolul 7. Este destul să spunem, deocamdată, că în anumite cazuri, unele tabele există pentru a oferi un punct de pornire operațiilor de introducere a datelor în alte tabele. Nu uitați, după ce un utilizator alege un tip de activitate, tariful acesteia se poate schimba.

Tabelul Invoice

În final, dar nu mai puțin important, vom discuta despre tabelul Invoice. Acesta este foarte simplu și are structura:

- InvoiceID (identificatorul facturii) – cheie primară;
- ClientID (identificatorul clientului) – cheie externă de la tabelul Client;
- Invoce Number (numărul facturii);
- Invoice Date (data facturii).

În acest context, tabelul Invoice nu este decât un „adeziv" pentru a uni mai multe înregistrări din tabelul TimeEntryDetail. Există un element din tabelul Invoice care îl face unic – câmpul Invoice Number. În acest context, numărul unei facturi este parțial static și parțial secvențial. De exemplu factura 150 din anul 2001 ar putea avea numărul 2001-150. În această situație are prefixul static iar restul este secvențial. Cum este produs numărul unei facturi? Întrucât ne aflăm încă în faza de modelare, nu ne vom preocupa de acest aspect.

Recapitulare a modelului bazei de date

Categoric, este momentul să luați o pauză! Figura 4.6 prezintă ultima versiune a modelului bazei de date Time Entry and Billing (TEB). Acesta este foarte diferit și mai expansiv decât cel realizat în capitolul 3. Se spune că o imagine valorează mai mult decât o mie de cuvinte. Figura 4.6 oferă o

bună înțelegere a funcțiilor pe care baza de date TEB le va suporta. Este important să observați că nu există încă o bază de date fizică. În realitate, veți prezenta acest model utilizatorilor primari ai sistemului pentru a vă asigura că au fost îndeplinite cerințele lor. Din perspectiva bugetului, acesta este momentul în care doriți să sesizați orice eventuale disfuncționalități. Dacă am compara o aplicație cu o construcție, atunci baza de date ar fi fundația. Dacă fundația este șubredă, construcția se va prăbuși. Nu veți putea începe construcția înainte de a turna fundația și înainte ca ea să se fi uscat. Dacă vă grăbiți, nu vă așteaptă lucruri bune. În procesul de dezvoltare, acesta este momentul în care trebuie să vă asigurați că fundația aplicației dumneavoastră s-a uscat.

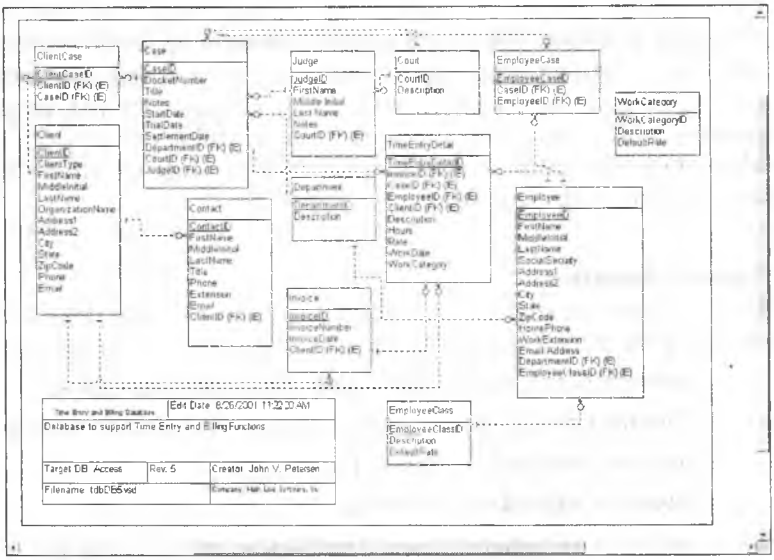


FIGURA 4.6
Ultima versiune a bazei de date Time Entry and Billing a fost completată cu noi coloane și tabele și a parcurs procesul de normalizare.

PUNEȚI ÎNTREBĂRI ȘI IMPLICAȚI UTILIZATORII!

O metodă prin care puteți stabili că proiectul dumneavoastră este solid este să puneți întrebări bazei de date și să stabiliți dacă vă poate oferi răspunsuri. De exemplu, puneți întrebarea „Se poate determina care avocați lucrează la un anumit caz?", apoi vedeți dacă baza de date poate răspunde la această întrebare. În acest caz, poate. În analiza designului, vedeți o legătură între cazuri și angajați. Mai mult angajații sunt grupați pe categorii în tabelul EmployeeClass. Evident, dacă puneți o întrebare la care baza de date nu poate răspunde, designul acesteia nu este complet.

Așa cum bănuiți, probabil, observarea acestor lipsuri acum, în faza de modelare, este mult economicoasă. Existența lipsurilor poate fi determinată prin instinct, nu numai logic. În mod sigur, știința nu poate răspunde tuturor întrebărilor. Dacă nu știți prea bine ce întrebări să puneți, aflați ce doresc să știe utilizatorii sistemului. Nu uitați să solicitați întrebări de la utilizatorii aflați pe toate treptele ierarhice. Vă mai amintiți discuția noastră anterioară despre nevoile tactice și nevoile strategice ale unei firme? Utilizatorii vor aprecia în mod sigur faptul că au fost implicați în acest proces. Cu cât ați implicat mai mulți utilizatori, cu atât este mai probabil ca aceștia să preia și să accepte sistemul.

Rezumat

În acest capitol, ați fost introdus în concepetele normalizării și ale impactului acesteia asupra designului bazei de date. În majoritatea situațiilor, o bază de date trebuie să se afle numai în a treia formă normală. În unele situații, totuși, este necesară aducerea la a patra și a cincea formă normală. Pe măsură ce parcurgeți procesul de normalizare, veți putea stabili ce coloane trebuie incluse în ce tabele. Este aproape sigur că eforturile dumneavoastră de normalizare vă vor conduce la crearea unor tabele la care nu vă gândiserăți în eforturile inițiale de proiectare. Acest capitol și cel precedent au schițat o hartă și un cadru pentru efectuarea analizei și proiectării bazei de date. Analiza și proiectarea bazei de date pot fi complexe, deci scopul acestor două capitole a fost de a sublinia simplitatea și bunul simț. Recapitulând, iată cadrul pe care l-am creat:

- Adunați informații despre cerințele pe care trebuie să le îndeplinească baza de date, folosind interviurile, documentele firmei și sistemele existente.
- Din aceste informații, identificați aspectele importante pe care trebuie să le urmărească baza dumneavoastră de date. Aceste „aspecte" vor lua forma tabelelor.
- După identificarea tabelelor importante, identificați relațiile dintre ele. Din acest proces pot rezulta noi tabele.
- După ce ați pus la punct elementele primare, aveți un cadru pe care puteți construi. Utilizarea normalizării și a informațiilor adunate anterior, este posibil să vă conducă la identificarea de noi câmpuri de date și tabele.
- În final, întrebați modelul bazei de date ceea ce firma dorește să afle. Acest proces determină completitudinea modelului. Dacă modelul nu poate răspunde la o întrebare, va trebui să repetați unii sau toți pașii precedenți.

Aici se încheie studiul analizai și proiectării unei baze de date în sens formal. Acum vom trece la aspecte mult mai tangibile: construirea și punerea în funcțiune a bazei de date. Capitolul 5 vă asistă în procesul construirii în Access a bazei de date Time Entry and Billing (TEB).



5

CONSTRUIREA BAZEI DE DATE ÎN ACCESS

În acest capitol:

- Generarea automată a bazei de date din modelul TEB;
- O scurtă recapitulare a utilitarului Access Database Designer;
- Crearea bazei de date folosind utilitarul Access Database Designer;
- Rezumat

Acum, după ce ați încheiat modelul bazei de date, este momentul să creați baza de date fizică. Există două metode pentru a crea o bază de date. Cu prima metodă, veți crea automat baza de date din diagrama Visio. A doua metodă presupune crearea automată a tuturor tabelelor, câmpurilor și relațiilor. Chiar dacă alegeți metoda automată, tot va trebui să configurați manual câteva atribute ale câmpurilor. Acesta este motivul pentru care acest capitol se va concentra asupra metodei manuale. Capitolul va începe cu metoda manuală, pentru cazul în care alegeți această cale. Figura 5.1 ilustrează punctul dumneavoastră de pornire, modelul bazei de date Time Entry and Billing (TEB), pe care ați creat-o în capitolul 4.

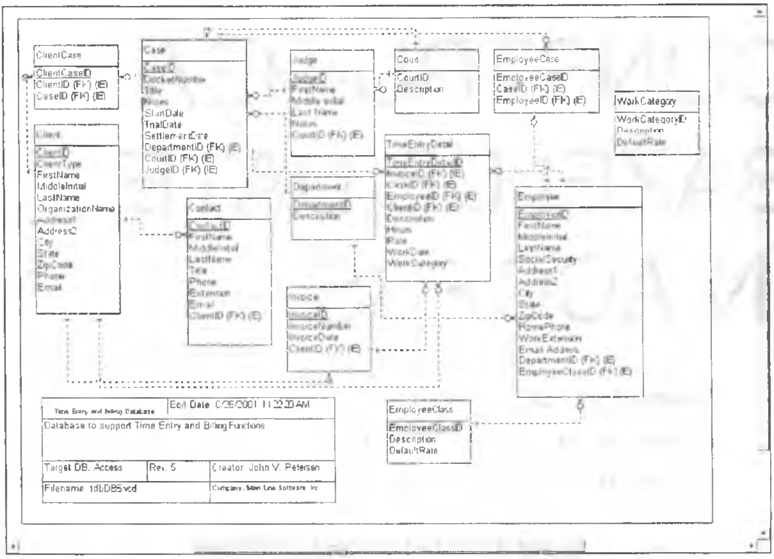


FIGURA 5.1
Modelul bazei de date Time Entry and Billing.

Generarea automată a bazei de date din modelul TEB

În capitolul 3 am prezentat o introducere în aplicația wizard Create Database din Visio. Aceasta citește o diagramă cu relații între entități din Visio și adaugă tabele într-o bază de date existentă sau creează o nouă bază de date. Pentru detalii despre funcționarea funcțiilor din Visio, opriți-vă puțin și reveniți la capitolul 3. În pașii următori vom recapitula pe scurt modul de utilizare a aplicației wizard.

După ce deschideți modelul bazei de date în Visio, din meniul Tools, alegeți Database Design\Create Database Wizard. Figura 5.2 prezintă caseta de dialog a aplicației wizard.

FIGURA 5.2

Ecranul de deschidere al aplicației wizard Create Database.

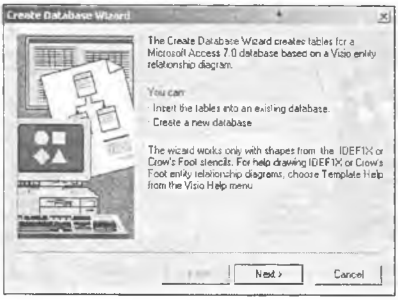
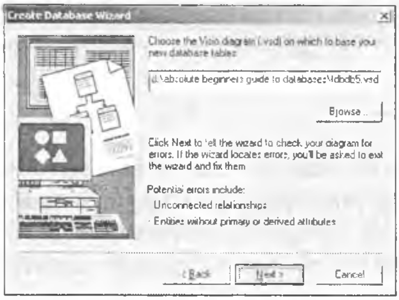


Figura 5.3 prezintă pasul 2 al aplicației, care vă cere să specificați numele fișierului cu diagrame din Visio (.vsd) pe care îl veți utiliza pentru crearea bazei de date.

FIGURA 5.3

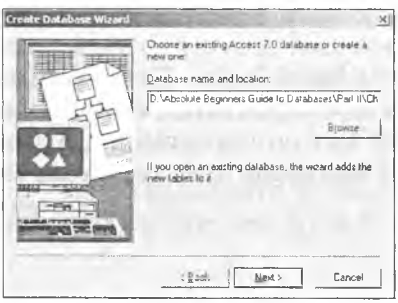
Pasul 2 al aplicației wizard vă solicită numele fișierului din Visio pe care îl veți folosi.



Între pașii 2 și 3, Visio va citi fișierul. În pasul 3 veți indica programului Visio locația în care va fi creat noul fișier cu baza de date. În acest caz, denumiți fișierul TEB.MDB (vedeți figura 5.4).

FIGURA 5.4

Pasul 3 al aplicației vă solicită numele bazei de date care va fi creată.



La ultimul pas, este afișată lista tabelelor care vor fi create. După ce executați clic pe butonul Finish, va fi creată noua bază de date. Figura 5.5 prezintă acest pas.

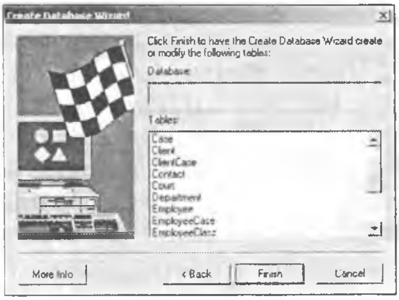


FIGURA 5.5

La ultimul pas este afișată lista tabelelor care vor fi create.

După crearea bazei de date, executați dublu clic pe pictograma din figura 5.6. Va fi lansat programul Access și va fi deschisă noua bază de date Time Entry and Billing.



FIGURA 5.6

Pictograma noii baze de date, afișată în Windows Explorer.

Figura 5.7 prezintă sistemul bazei de date Time Entry and Billing în Access. Diagrama relațiilor arată la început foarte dezordonat. În câteva minute, dacă mutați titlurile, puteți face puțină curățenie. Acum, după ce ați scăpat de dezordine, toate relațiile specificate în diagrama din Visio se află în actuala bază de date.

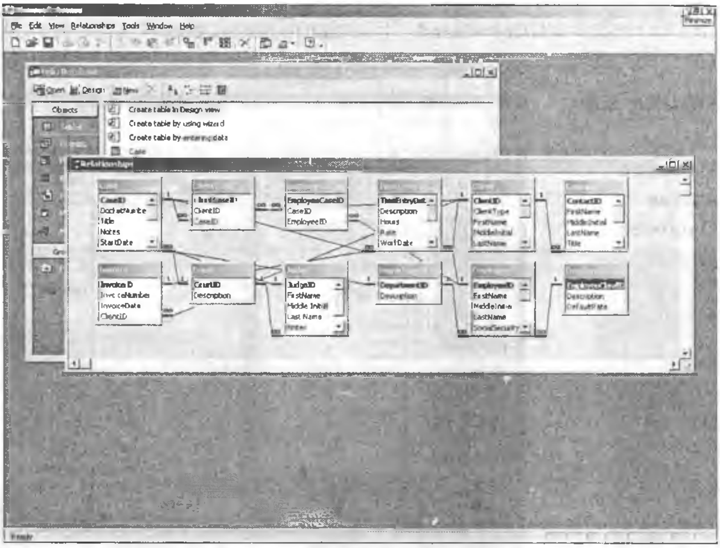


FIGURA 5.7

Relațiile definite în Visio sunt create în fișierul cu baza de date din Access.

Pe de o parte, această metodă de a crea baze de date în Access este foarte rapidă și eficientă. Pe de altă parte, nu veți înțelege cum sunt create de fapt aceste elemente dacă nu cunoașteți procesul. Până la urmă, aceasta este metoda prin care veți crea versiunile inițiale ale bazelor dumneavoastră de date. Întrucât această carte prezintă detaliile construirii și lucrului cu bazele de date, restul capitolului se va concentra pe metodele manuale de creare a tabelelor, coloanelor și relațiilor.

O scurtă recapitulare a programului Access Database Designer

În capitolul 2, am făcut o introducere în Access Database Designer. În acest capitol veți lucra cu acest program. Înainte de a trece mai departe, să analizăm puțin sitauția, utilizând ca exemplu baza de date Northwind Trader's. Figura 5.8 prezintă principalele componente din Access Database Designer.

Secțiunile următoare schițează și descriu pe scurt componentele principale ale designului bazei de date cu care veți lucra în acest capitol.

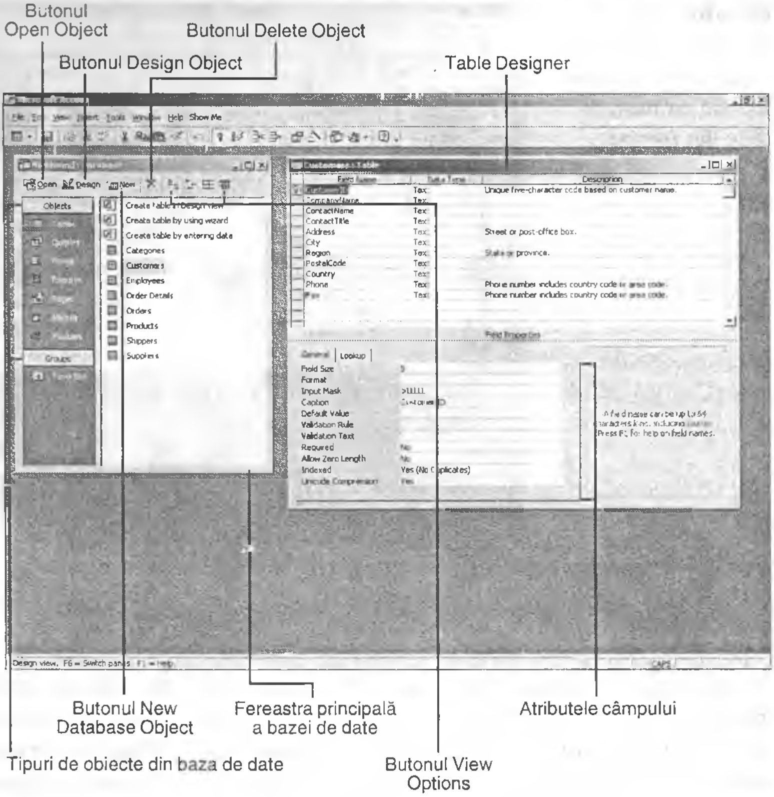


FIGURA 5.8
Interfața Microsoft Access Database Design.

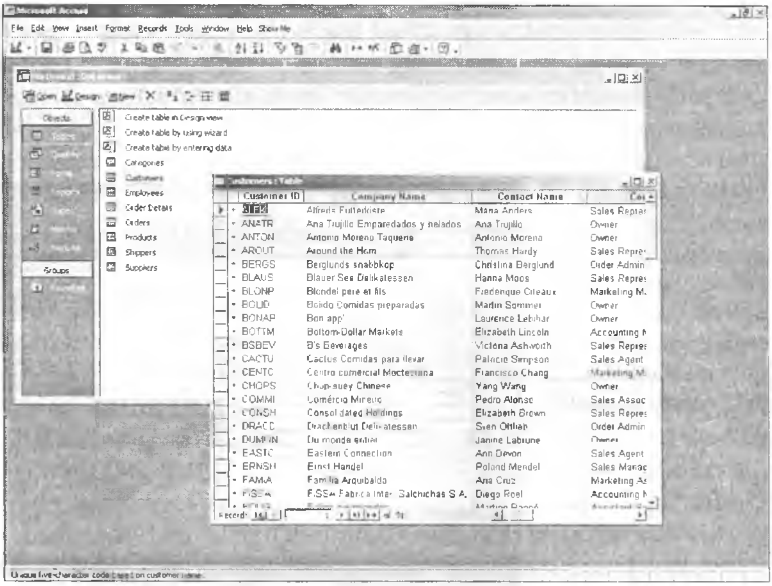
Fereastra principală a bazei de date

În figura 5.8 sunt afișate două ferestre. În stânga se află fereastra principală a bazei de date, care oferă acces la toate obiectele dintr-o bază de date. În deapta se află utilitarul de proiectare a tabelelor (***table designer***), despre care vom discuta mai târziu în acest capitol. Fereastra principală a bazei de date este și ea divizată în două panouri verticale. Panoul din stânga conține diferite obiecte și vă permite să selectați obiecte de diferite tipuri. Lista include tabele, interogări, formulare, rapoarte, pagini, macroinstrucțiuni și module. Când selectați un alt obiect în panoul din stânga, conținutul panoului din dreapta se modifică pentru a prezenta obiectele de tipul respectiv. Acest capitol se va concentra pe primul tip de obiecte, tabelele. Celelalte tipuri de obiecte vor fi discutate în capitolele următoare.

În partea de sus a ferestrei principale a bazei de date se află o bară de instrumente alcătuită din trei secțiuni. Bara de operații (***operations***) funcționează în contextul obiectului curent selectat. Dacă privim spre dreapta, prima secțiune a barei de instrumente are trei opțiuni: Open (deschidere), Design (proiectare) și New (obiect nou).

Efectul butonului Open este de „rulare (lansare)" a obiectului selectat. În cazul unui tabel, aceasta înseamnă navigarea prin datele acestuia. Figura 5.9 prezintă datele din tabelul Northwind Customers.

FIGURA 5.9
Butonul Open din fereastra principală a bazei de date rulează obiectul selectat.



Butonul Design plasează obiectul curent selectat în modul Design. Executarea unui dublu clic pe obiect în panoul din dreapta are același efect. Figura 5.8 prezintă tabelul cu clienți în modul design. Componentele utilitarului Table Designer sunt discutate pe larg în capitolul 2. Butonul New creează un obiect nou, pe baza tipului selectat în panoul din stânga al ferestrei principale a bazei de date.

Dacă executați clic pe butonul New, este afișată caseta de dialog New Table. Figura 5.10 prezintă diferitele opțiuni disponibile atunci când creați un tabel. Așa cum veți vedea, Access are multe aplicații wizard, care vă ghidează pas cu pas pentru îndeplinirea multor sarcini uzuale.

A doua secțiune a ferestrei principale a bazei de date conține o singură opțiune, care șterge obiectul curent selectat. A treia secțiune conține patru opțiuni, care controlează aspectul panoului din dreapta al ferestrei principale a bazei de date. Cele patru opțiuni sunt: Large Icons (pictograme mari), Small Icons (pictograme mici), List (listă) și Details (detalii). Figura 5.11 prezintă nivelul de detaliere a obiectelor pe care le puteți vedea.

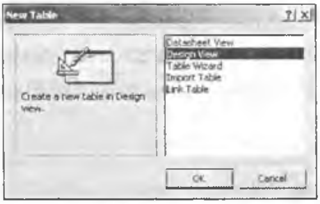


FIGURA 5.10

Caseta de dialog New Table prezintă mai multe metode de creare a unui tabel.

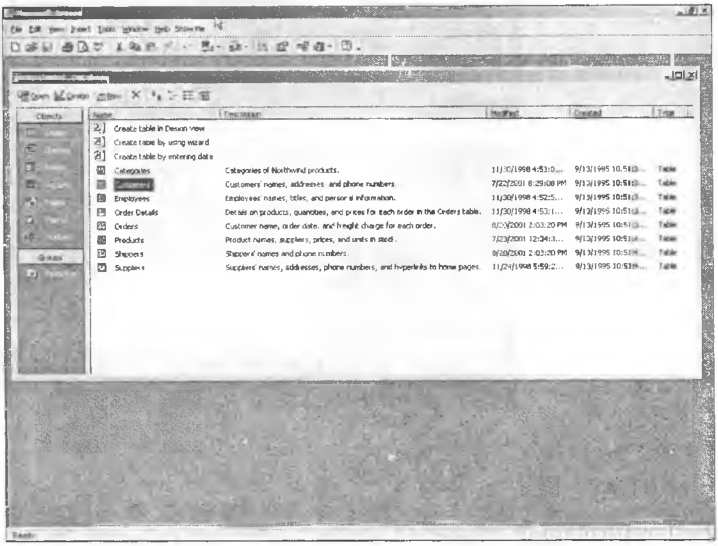


FIGURA 5.11

Detaliile de vizualizare pentru fiecare obiect al bazei de date conțin descrierea, data ultimei modificări, data creării și tipul.

Fereastra Designer

Să ne îndreptăm acum atenția asupra celei de a doua ferestre prezentate în figura 5.8, fereastra Designer. În Access există mai multe utilitare de design. Acestea sunt destinate tabelelor, interogărilor, formularelor, rapoartelor, paginilor web, macroinstrucțiunilor și modulelor. Utilitarul Table Designer a fost prezentat în capitolul 2 și va reprezenta subiectul acestui capitol. Vom prezenta și alte utilitare când vom discuta despre tipurile respective de obiecte.

Acum, după ce cunoaștem în mare situația, să trecem la crearea bazei de date Time Entry and Billing.

Crearea bazei de date folosind utilitarul Design

Această secțiune vă va explica pas cu pas cum să creați tabelele, coloanele și relațiile. Capitolul 2 v-a prezentat instrumentele de proiectare a bazelor de date din Access. Acum veți vedea și veți experimenta funcționarea acestor instrumente într-o situație reală. Procesul va începe cu crearea unei baze de date fără conținut. După crearea ei, veți crea tabelele. În fiecare tabel, veți defini coloanele, atributele fiecărei coloane și cheia primară. După ce construiți baza de date, veți defini relațiile dintre tabele. Iar după îndeplinirea acestor sarcini, veți avea la dispoziție elementele esențiale ale bazei de date Time Entry and Billing. Atunci veți fi gata de a trece la pasul următor, punerea în funcțiune a bazei de date.

Crearea bazei de date

Crearea bazei de date este cel mai ușor dintre pași. Din meniul File, selectați New. Se deschide caseta de dialog New, care conține două tab-uri, General și Databases (vedeți figura 5.12). În tab-ul General, puteți crea o bază de date fără conținut. Acesta vă oferă și posibilitatea de a crea o bază de date pornind de la un template (șablon) – vedeți figura 5.13. Cu cât veți dezvolta mai multe baze de date, veți descoperi că acestea se împart în funcție de diferitele tipuri de activități ale firmei. Bazele de date dintr-o anumită categorie au multe însușiri comune.

FIGURA 5.12

Tab-ul General din caseta de dialog New vă oferă posibilitatea de a crea o bază de date fără conținut.

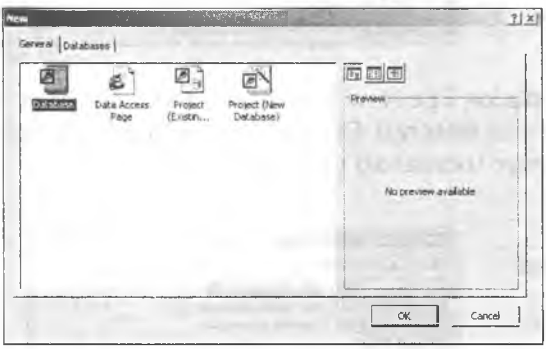
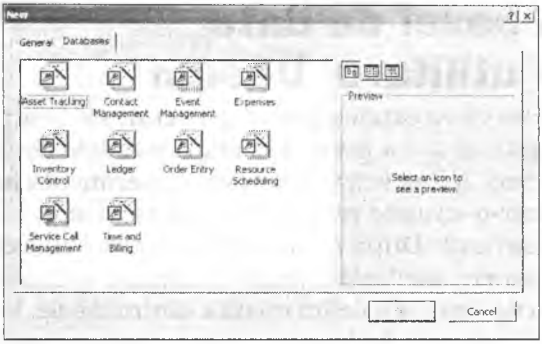


FIGURA 5.13

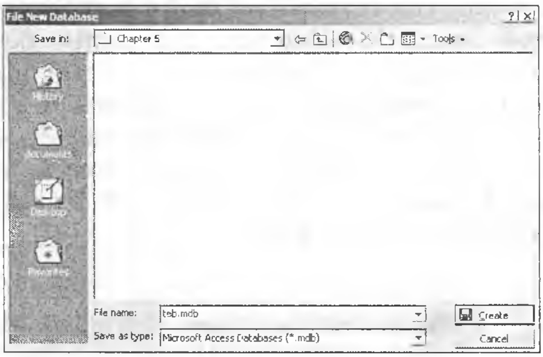
Tab-ul Databases din caseta de dialog New vă oferă posibilitatea de a crea o bază de date pe baza unui model (template) predefinit.



Din tab-ul General, selectați pictograma Database și selectați butonul OK. Va fi afișată caseta de dialog File New Database, specificând locația și numele noii baze de date (vedeți figura 5.14).

FIGURA 5.14

Caseta de dialog File New Database oferă posibilitatea de a specifica locația și numele noului fișier cu baza de date.



În caseta de dialog File New Database, denumiți baza de date TEB.MDB și executați clic pe butonul Create. Figura 5.15 prezintă baza de date fără conținut pe care tocmai ați creat-o.

FIGURA 5.15

Baza de date TEB fără conținut, pregătită pentru crearea noilor tabele, coloane și relații.



Indiciu

Poate ați observat că atunci când deschideți baza de date TEB, reprezentată de fișierul teb.mdb, apare fișierul teb.ldb. Figura 5.16 prezintă acest fișier. El facilitează accesul mai multor utilizatori (accesul multiutilizator) la baza de date. Deși nu sunteți familiarizat cu conceptul de multiutilizator, merită să vă opriți acum puțin pentru a atinge subiectul. Gândiți-vă la cazul în care un utilizator lucrează cu datele, iar altul dorește să le editeze. Cum rezolvă baza de date această problemă? Aici intervine fișierul ldb. El conține informații despre înregistrările blocate și utilizatorii care le pot debloca.

Pentru mai multe informații despre fișierele ldb, consultați Microsoft Developer Network Knowledge Base (MSDN KB). Dacă nu cunoașteți MSDN și baza sa de informații, categoric va trebui să devină o parte importantă a arsenalului dumneavoastră. Site-ul web MSDN poate fi accesat la `http://msdn.microsoft.com`. Baza de informații poate fi accesată la `http://search.support.microsoft.com/kb/`. Identificatorul specific al articolului despre fișierele ldb este Q208778.

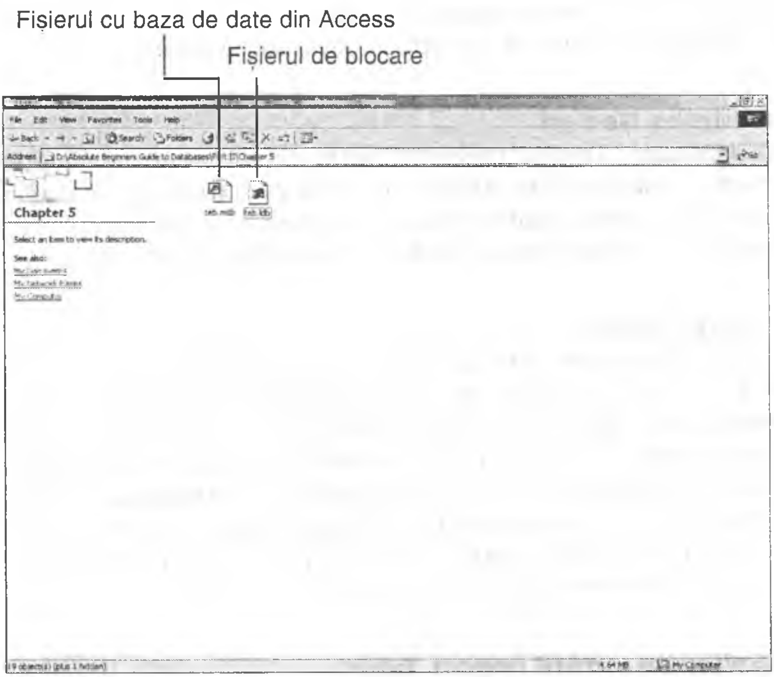


FIGURA 5.16 Problemele de utilizare multiplă sunt administrate cu un fișier de blocare, care are același nume ca și baza de date.

După crearea bazei de date, este momentul să creați tabelele și coloanele pentru baza de date Time Entry and Billing.

Crearea tabelelor și coloanelor

Crearea tabelelor și coloanelor în Access este foarte simplă. Primul pas este de a crea un nou tabel. Pentru aceasta, după ca ați selectat tipul de obiecte Tables în panoul din stânga, apăsați butonul New din bara de instrumente a ferestrei principale a bazei de date. Revedeți figura 5.8 pentru detalii despre fereastra principală a bazei de date. După ce executați clic pe butonul New, veți întâlni caseta de dialog New Table (revedeți figura 5.10). În acest caz, veți crea un tabel în vizualizarea Design. Mai mult ca sigur, veți utiliza vizualizarea Design aproape tot timpul. Aveți la dispoziție mai mule tabele noi.

Vizualizarea Datasheet

Dintre toate opțiunile aceasta este cea mai puțin utilă. Dacă vă place să creați tabelele și, în același timp, să introduceți datele, această abordare vă va plăcea. Veți începe cu ceva asemănător unei foi de date, conținând 10 coloane. Puteți redenumi câmpurile dacă executați clic cu butonul drept al mouse-ului pe numele câmpului și scrieți noul nume. Nu aveți controlul asupra altor atribute ale câmpurilor, cum ar fi tipul de date, mască de intrare, descriere și așa mai departe. Pentru a obține controlul asupra acestor elemente, trebuie să treceți în vizualizarea Design.

Vederea Design

Vizualizarea Design, prezentată în figura 5.8, este metoda preferată de proiectare a tabelelor. Vizualizarea este divizată în două părți. Partea de sus este formată din informațiile de bază despre câmp. Partea de jos conține atributele detaliate despre câmpul selectat curent în partea de sus a vizualizării.

Table Wizard

Dacă vă amintiți, puteți crea o bază de date pornind de la un model predefinit. Chiar dacă nu alegeți această cale pentru crearea bazei de date, tot aveți acces la șabloanele tabelelor prin aplicația Table Wizard. Aplicațiile wizard sunt utile, dar tot mai este nevoie să învățați și să înțelegeți funcționarea acestor elemente de bază. Pe măsură ce câștigați mai multă experiență, veți descoperi că aplicațiile wizard lungesc timpul de efectuare a unei operații. Așa cum veți vedea foarte repede, la fel ca toate regulile, nici aceasta privind aplicațiile wizard nu este absolută.

Aplicația wizard Import Table

Această aplicație vă ghidează prin procesul de preluare a definiției unui tabel dintr-o sursă externă, cum ar fi o altă bază de date din Access sau o altă bază de date orientată obiect și utilizarea sa ca bază pentru un nou tabel în baza dumneavoastră de date din Access. În acest scenariu, aplicația wizard realizează ușor ceea ce altfel ar fi o sarcină complexă. În astfel de cazuri, aplicațiile wizard se dovedesc foarte utile.

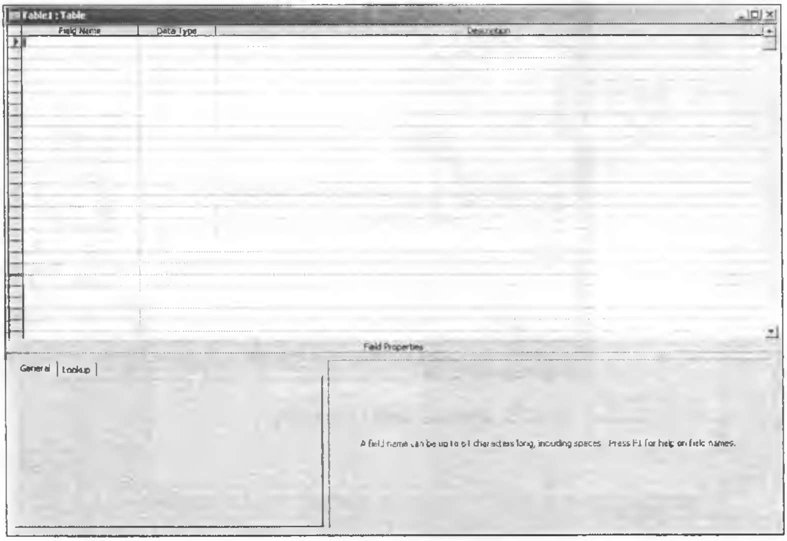
Aplicația wizard Link Table

Aplicația wizard Link Table este asemănătoare cu Table Wizard, cu o singură excepție notabilă. Definiția tabelului pe care îl creați în Access face legătura cu o sursă externă de date. De fapt, această sursă externă stochează datele. Continuând lucrul la noul tabel, în caseta de dialog New table, selectați vizualizarea Design și executați clic pe butonul OK.

Va apărea proiectul unui tabel liber, în care puteți crea câmpuri (vedeți figura 5.17).

FIGURA 5.17

Iată cum apare proiectul unui tabel nou creat.



Nu puteți salva un tabel dacă nu ați creat cel puțin un câmp. Orice tabel trebuie să aibă o cheie primară. Primul câmp pe care îl veți crea va fi cheia primară. Revenind la modelul bazei de date, câmpul cheie primară pentru tabelul Client se numește ClientID. Figura 5.18 prezintă câmpul nou creat.

Crearea unui câmp este ușoară. Urmați pașii:

1. Scrieți numele câmpului în coloana Field Name (`ClientID`).
2. Atribuiți un câmp de date în coloana Data Type (`AutoNumber`).
3. Scrieți o descriere în coloana Description (Primary Key field for Client table – nu este obligatoriu ca textul scris în această coloană să fie în limba engleză, n. tr.).

Tipul de date `AutoNumber` este utilizat pentru cheile primare. Access ține evidența următorului identificator secvențial care va fi utilizat pentru noile înregistrări. Mai este ceva de făcut. Trebuie să desemnați câmpul ClientID ca fiind cheie primară. Pentru asta, este suficient să apăsați butonul Primary Key, prezentat în figura 5.18. Obsevați indicatorul Primary Key, care apare în stânga numelui câmpului.

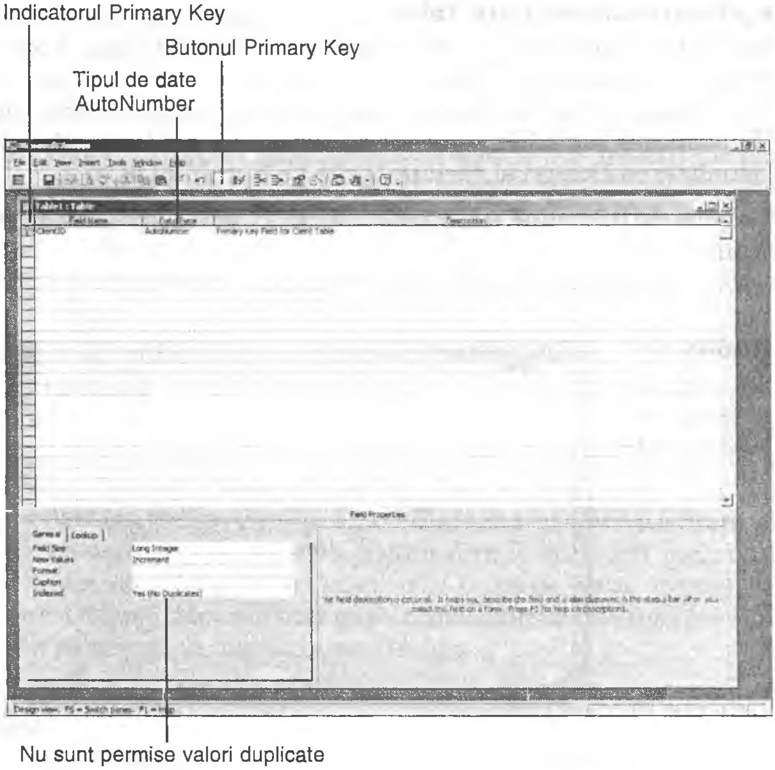


FIGURA 5.18
Înainte da a salva un tabel, trebuie să creați cel puțin un câmp și să definiți cheia primară.

După ce ați setat cheia primară, este bine să salvați imediat tabelul. Puteți face acest lucru executând clic pe butonul Save de pe bara principală de instrumente din Access sau din meniul File puteți selecta Save. Orice metodă ați alege, va apărea caseta de dialog Save As (vedeți figura 5.19). în acest caz, denumiți tabelul „client" și apăsați butonul OK.

Figura 5.20 prezintă noul tabel Client, care acum face parte din baza de date TEB.

Indiciu

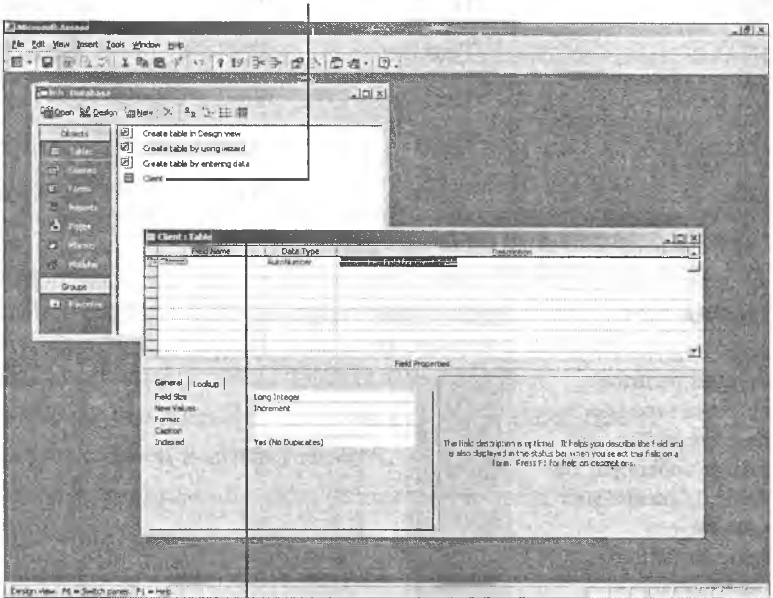
Dar dacă doriți să modificați numele noului obiect? Modificarea numelor obiectelor este ușoară. Prima regulă este că obiectul nu poate fi deschis când îi modificați numele. Când sunteți gata, selectați obiectul în fereastra principală a bazei de date și executați clic cu butonul drept al mouse-ului. Figura 5.21 prezintă meniul dependent de context care apare (numit meniu contextual, meniu instantaneu sau meniu pop-up). Selectați Rename. Cursorul va fi poziționat la începutul numelui, oferindu-vă posibilitatea de a scrie un nume nou. Altfel, puteți selecta obiectul în fereastra principală a bazei de date, apoi să executați un clic pe numele său.



FIGURA 5.19

Caseta de dialog Save As vă permite să specificați numele noului obiect al bazei de date.

Noul tabel al bazei de date, denumit Client



Numele tabelului este prezentat în fereastra de proiectare

FIGURA 5.20

După salvarea unui obiect al bazei de date, acesta apare imediat în fereastra principală a bazei de date.

După ce ați învățat să creați un câmp, ceea ce veți face este repetarea procesului. Tabelul 5.1 prezintă detaliile specifice fiecărui câmp.

Tabelul 5.1 Informații despre câmpurile tabelului Client

Nume	Tip	Dimensiune	Descriere
`ClientID`	AutoNumber	LI*	Cheie primară a tabelului Client
`ClientType`	Number	LI	Cheie străină de la tabelul Client Type
`FirstName`	Text	30	Prenumele
`MiddleInitial`	Text	1	Inițiala celui de-al doilea nume

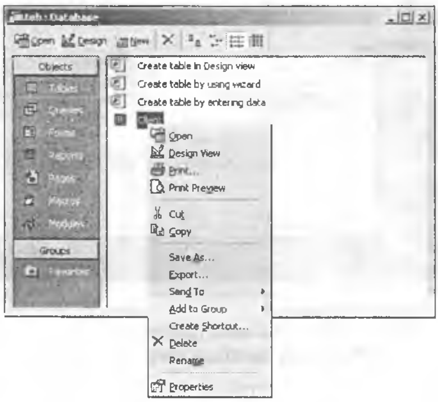
Tabelul 5.1 (continuare)

Nume	Tip	Dimensiune	Descriere
`LastName`	Text	30	Numele de familie
`OrganizationName`	Text	50	Numele organizației
`Address1`	Text	50	Adresa 1
`Address2`	Text	50	Adresa 2
`City`	Text	30	Orașul
`State`	Text	2	Statul
`ZipCode`	Text	9	Codul poștal
`Phone`	Text	10	Numărul de telefon
`Email`	Text	50	Adresa de e-mail

LI este prescurtarea de la Long Integer.

FIGURA 5.21

Dacă executați clic dreapta pe orice obiect al bazei de date, va apărea un meniu dependent de context, conținând opțiuni corespunzătoare obiectului.



Notă

Când introduceți informații despre câmpuri, veți vedea un set de atribute ale acestora (de exemplu, dacă este obligatorie completarea unui câmp sau dacă poate fi lăsat necompletat). Nu vă preocupați acum de setările pentru aceste proprietăți suplimentare. În capitolele următoare vom realiza o prezentare introductivă a acestor proprietăți și atunci veți învăța care este rolul lor. În acest moment, vă veți ocupa numai de proprietățile:

- Nume;
- Tip;
- Descriere;
- Dimensiune.

După crearea câmpurilor menționate anterior, comparați tabelul dumneavoastră cu cel din figura 5.22.

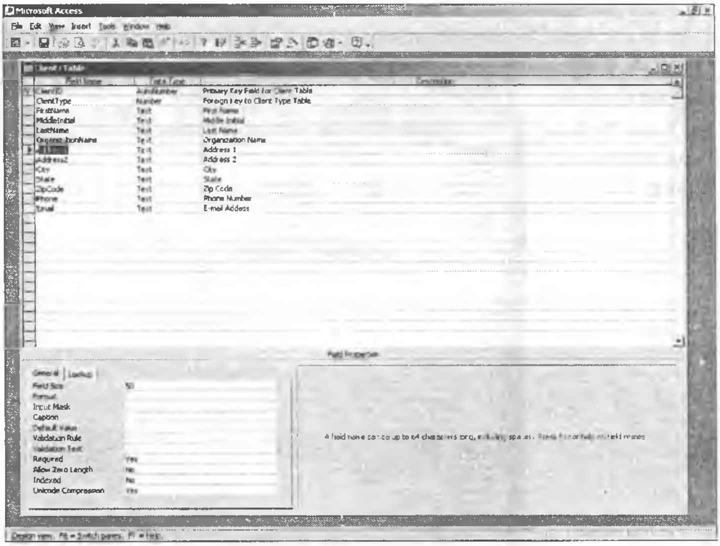


FIGURA 5.22

Tabelul Client din baza de date Time Entry and Billing.

Pentru a înțelege modul de construire a relațiilor, va trebui să aveți la dispoziție cel puțin două tabele. Folosind informațiile din tabelul 5.2, construiți tabelul Contact. Priviți figura 5.23 pentru a verifica ceea ce ați realizat. Nu uitați să efectuați pașii următori:

1. Executați clic pe butonul New pentru a crea un tabel nou.
2. Creați câmpul care va fi cheie primară.
3. Atribuiți cheia primară câmpului pe care l-ați creat la pasul 2.
4. Salvați tabelul în baza de date.

Tabelul 5.2 Informații despre câmpurile tabelului Contact

Nume	Tip	Dimensiune	Descriere
`ContactID`	Autonumber	LI	Cheie primară pentru tabelul Client
`ClientID`	Number	LI	Cheie străină de la tabelul Client
`FirstName`	Text	30	Prenumele
`MiddleInitial`	Text	1	Inițiala celui de-al doilea nume
`LastName`	Text	30	Numele de familie
`Title`	Text	30	Titlul
`Phone`	Text	10	Numărul de telefon
`Extension`	Text	5	Interiorul
`Email`	Text	50	Adresa de e-mail

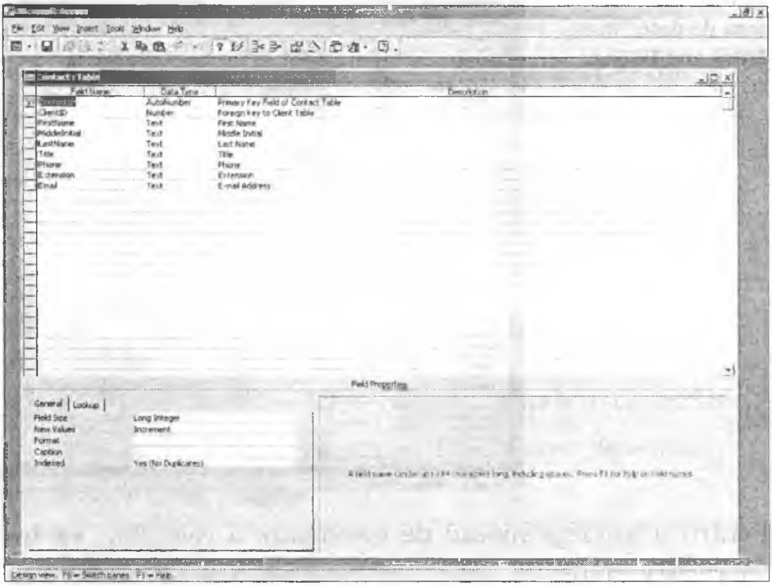


FIGURA 5.23
Tabelul Contact din baza de date Time Entry and Billing.

În acest moment, poate doriți să creați restul bazei de date TEB. Structura detaliată a fiecărui tabel este prezentată în anexa A. o altă variantă este să așteptați până la sfârșitul acestui capitol, deoarece secțiunea următoare, al cărei subiect sunt relațiile, utilizează numai tabelele Client și Contact.

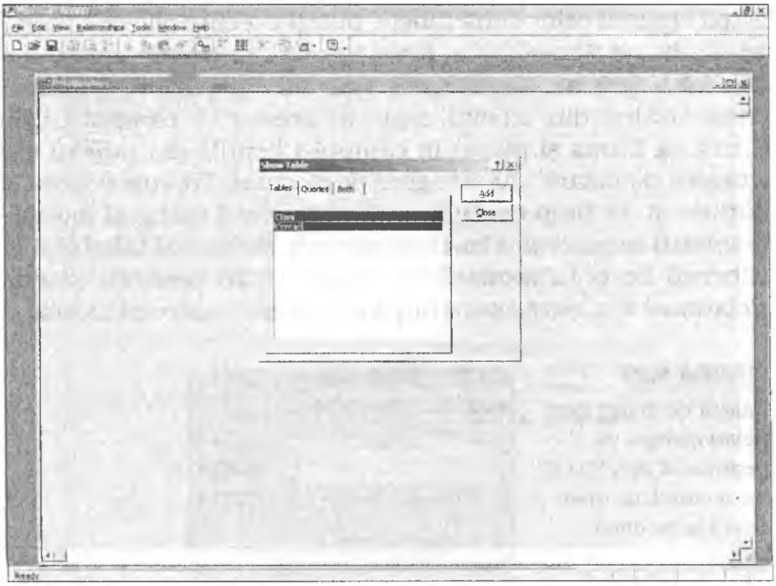
Crearea relațiilor

Având la dispoziție două tabele, puteți construi o relație între tabelele Client și Contact. Dacă revedeți modelul, veți observa că există o relație unu la mulți (1:M) între cele două tabele. Aceasta înseamnă că pentru un client pot exista mai multe contacte. După ce ați aflat care sunt caracteristicile relației, pasul următor este crearea relației în baza de date din Access.

Din meniul Tools, selectați Relationships. Figura 5.24 prezintă aplicația de construcție a relațiilor și caseta de dialog Show Table care apare. Dacă nu apare automat caseta de dialog Show Table, selectați Show Table din meniul Relationships.

FIGURA 5.24

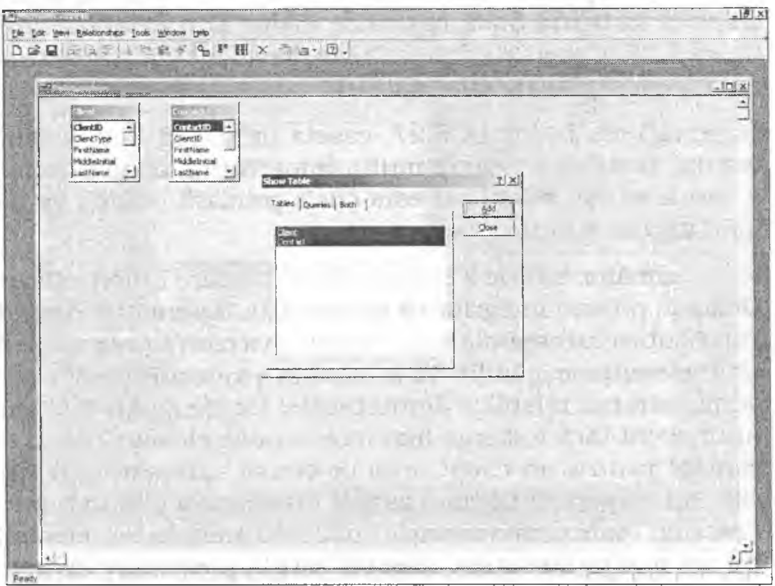
Aplicația de construire a relațiilor în bazele de date din Access.



În caseta de dialog Show Table din figura 5.24, asigurați-vă că sunt selectate tabelele Client și Contact. Puteți selecta o opțiune dintr-o listă ținând apăsată tasta Shift în timp ce execuatați clic cu mouse-ul pe opțiune. După selectarea tabelelor, executați clic pe butonul Add. Aplicația de construire a relațiilor este asemănătoare celei din figura 5.25.

FIGURA 5.25

Aplicația de construire a relațiilor din bazele de date în Access, care conține tabelele Client și Contact.



După crearea celor două tabele, puteți executa clic pe butonul Close al casetei de dialog Show Table. Există două metode de creare a relațiilor. Prima implică tragerea câmpului cheie din tabelul părinte și plasarea sa în câmpul cheie străină din tabelul copil. În acest caz, câmpul ClientID din tabelul Client va fi tras și plasat în câmpul ClientID din tabelul Contact. Operația „tragere și plasare" nu este greu de efectuat. Trebuie doar să selectați câmpul părinte și, în timp ce țineți apăsat butonul stâng al mouse-ului, să trageți pointerul mouse-ului în câmpul cheie străină al tabelului copil, iar apoi să eliberați butonul mouse-ului. Figura 5.26 prezintă caseta de dialog Edit Relationships, care apare după ce eliberați butonul mouse-ului.

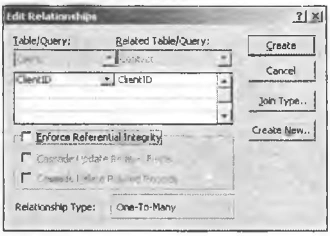


FIGURA 5.26

Caseta de dialog Edit Relationships vă permite să specificați caracteristicile unei relații între două tabele.

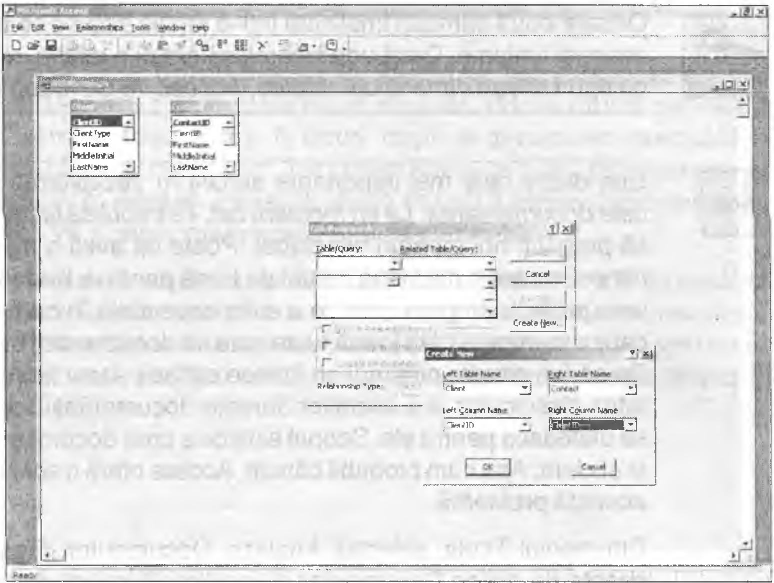
A doua metodă prin care puteți crea o relație este de a selecta opțiunea Edit Relationships din meniul Relationship. Va fi afișată aceeași casetă de dialog, Edit Relationships, pe care ați văzut-o și în figura 5.26. Singura diferență este că acum caseta de dialog este liberă. Va trebui să executați clic pe butonul Create Now pentru a afișa caseta de dialog Create New Relationships (vedeți figura 5.27).

După ce completați informațiile necesare în caseta de dialog New Relationships ca în figura 5.26, caseta de dialog Edit Relationships va arăta ca în figura 5.27. Caseta de dialog Create New Relationships înlocuiește operația de tragere și plasare cu mouse-ul.

Observați că, în figura 5.27, caseta de dialog Edit Relationships știe că relația creată este unu la mulți. Întotdeauna când câmpul aflat în partea dreaptă a unei relații nu este cheie primară, relația va fi considerată ca fiind unu la mulți.

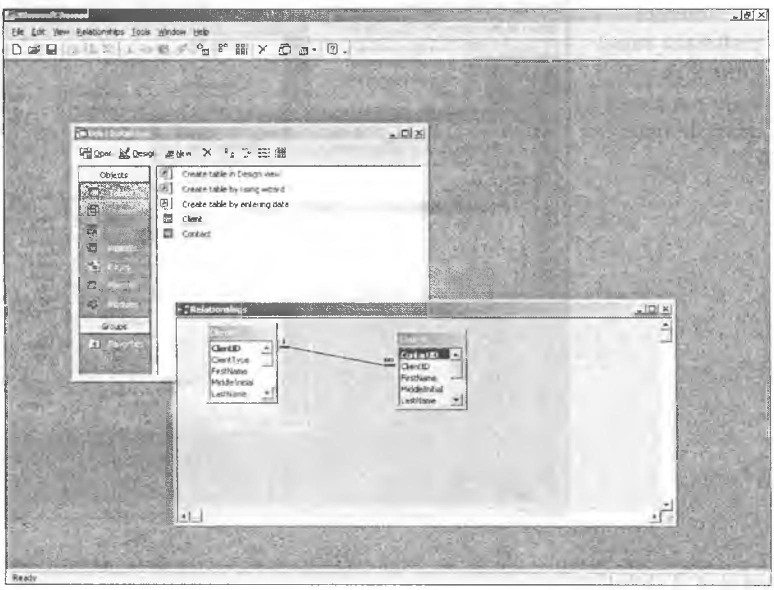
Pasul următor este de a bifa caseta de validare Enforce Referential Integrity. Detaliile privind integritatea referențială (***referential integrity*** – RI) vor constitui subiectul capitolului 7. Pentru a vă oferi o prezentare rapidă a integrității referențiale, gândiți-vă la ea ca la o indicație dată bazei de date pentru administrarea relațiilor dintre tabele. De exemplu, este permisă ștergerea unui client fără a șterge mai întâi contactele sale? Sau, dacă ștergeți un contact pentru un client, trebuie șterse automat toate contactele pentru clientul respectiv? Ultimul aspect este cunoscut sub numele de ***ștergeri în cascadă***. Toate aceste exemple ilustrează integritatea referențială, dar subiectul are mai multe fațete. Această scurtă prezentare ar trebui să vă ofere o bună înțelegere preliminară a motivelor pentru care bazele de date suportă integritatea referențială.

FIGURA 5.27
Caseta de dialog Create New Relationships vă permite să specificați manual caracteristicile unei relații.



După ce bifați caseta de validare Referential Integrity, executați clic pe butonul OK. Figura 5.28 prezintă relația nou creată. Observați numărul 1 aflat la unul dintre capetele relației și semnul infinitului, care seamănă cu un opt întors pe orizontală, la capătul „mulți" al relației.

FIGURA 5.28
Noua relație unu la mulți între tabelele Client și Contact în aplicația de construire a relațiilor.



Indiciu

Oricare două câmpuri implicate într-o relație trebuie să aibă același tip și aceeași lungime. Dacă veți încerca să creați o relație între două câmpuri cu tipuri și/sau dimensiuni diferite, Access va genera o eroare.

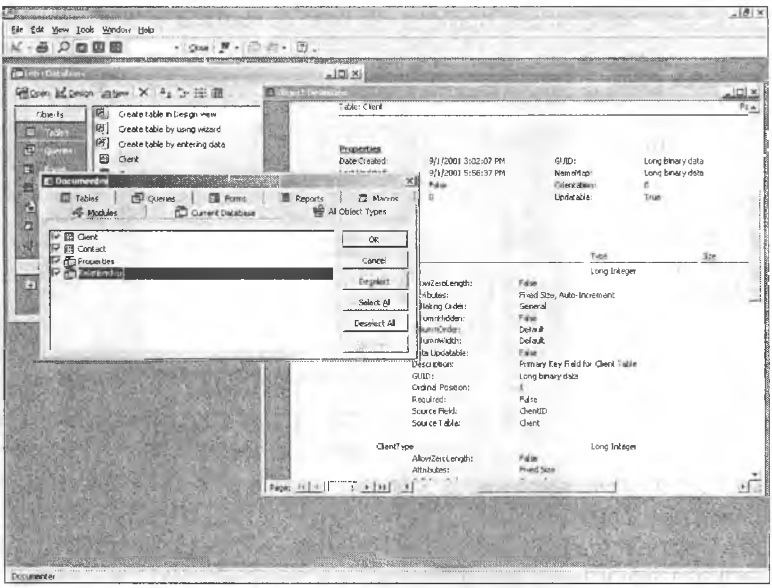
Notă

Una dintre cele mai importante sarcini în dezvoltarea bazelor de date este documentarea. La un moment dat, va trebui să faceți modificări și/sau să pregătiți noi membri ai echipei. Poate că aveți o memorie bună, dar nimeni nu are o memorie destul de bună pentru a memora toate detaliile unui proiect. Singura cale de a evita capcanele în care puteți cădea din cauza memoriei care joacă feste este să documentați bine baza de date. Ca să fim cinstiți, nimănui nu-i place să facă acest lucru. Și, deși majoritatea directorilor și a clienților dorește documentații, nimănui nu-i place să plătească pentru ele. Scopul este de a crea documente complete rapid și eficient. Așa cum probabil bănuiți, Access oferă o soluție drăguță pentru această problemă.

Din meniul Tools, selectați Analyze, Documenter. Figura 5.29 prezintă caseta de dialog Documenter și rezultatul furnizat de utilitarul de documentare. Toate obiectele bazei de date pot fi documentate. Documentația poate fi printată din Access sau exportată în Microsoft Word sau Excel.

FIGURA 5.29

Utilitarul Documenter realizează rapid munca de documentare a bazei de date.



Rezumat

În acest capitol, am făcut o introducere în modul de transformare a unui model în bază de date. Au fost expuse două metode. Prima dintre ele este o cale rapidă de citire a modelului. A doua implică generarea manuală a tabelelor, coloanelor și relațiilor. De asemenea, a fost prezentată și construirea relațiilor în Access. Pe viitor, după ce veți fi înțeles bine detaliile, metoda automată va fi mult mai rapidă și mai eficientă.

Partea a II-a a cărții se încheie cu acest capitol, al cărui scop îl reprezintă fazele de proiectare și construire a unei baze de date. În acest moment, baza de date conține numai elementele esențiale. În partea a III-a veți trece la nivelul următor, care implică inteligența de a susține regulile firmei și integritatea bazei de date.

PARTEA A III-A

ÎNTREȚINEREA BAZEI DE DATE – INTRODUCERE ÎN INTEGRITATEA BAZELOR DE DATE



6

MENȚINEREA INTEGRITĂȚII PRIN REGULI DE VALIDARE

În acest capitol:

- O prezentare generală a funcționării regulilor de validare în bazele de date
- Implementarea regulilor la nivel de câmp
- Implementarea regulilor la nivel de înregistrare
- Rezumat

În capitolul 5 ați încheiat procesul de construire a elementelor esențiale ale bazei de date Time Entry and Billing (TEB). Acestea sunt tabelele, câmpurile și relațiile și reprezintă minimul necesar pentru stocarea datelor. Marea problemă este acum modul în care controlați ce anume se stochează în baza de date? Ar trebui ca utilizatorul să poată introduce orice dorește? Ar putea el să salveze înregistrarea despre un client fără o adresă completă a acestuia? Ar trebui să existe niște metode de control? Dacă da, atunci cum vor fi ele implementate? Aceste întrebări fac parte din subiectul vast al integrității bazelor de date.

Majoritatea sistemelor de administrare a bazelor de date relaționale (RDBMS) au capacitatea de a menține integritatea. Integritatea bazelor de date este clasificată în reguli de validare a datelor și integritate referențială. Regulile de validare se ocupă de aspecte precum faptul că datele introduse într-un câmp sunt valide (acceptabile) sau faptul că pot fi salvate în formatul în care sunt introduse. De exemplu, dacă un utilizator nu a introdus orașul unui client, înregistrarea nu va putea fi salvată. Acest capitol tratează modul de definire a regulilor la nivel de câmp și înregistrare. Integritatea referențială are ca obiectiv menținerea relației logice dintre tabele. De exemplu, este permisă ștergerea înregistrării unui client dacă există o înregistrare cu informații de contact despre el? Integritatea referențială este discutată în detaliu în capitolul 7, iar acest capitol va ilustra aplicarea regulilor de validare la nivel de câmp și înregistrare utilizând tabelul Client din baza de date TEB.

Prezentare generală a regulilor de validare

Bazele de date suportă reguli de validare la două niveluri. Primul este reprezentat de câmpuri. Regulile de validare la nivel de câmp mai sunt denumite validări la nivel de câmp. De exemplu, dacă utilizatorul trebuie avertizat imediat ce a introdus date nevalide într-un câmp, va fi folosită o regulă de validare la nivel de câmp. Ați fost vreodată în situația de a nu putea părăsi un câmp înainte de a introduce în el date valide? Aceasta este una dintre situațiile când sunt aplicate regulile de validare la nivel de câmp.

Regulile de validare la nivel de câmp sunt criticate adesea deoarece au tendința de a întrerupe fluxul de lucru. Este supărător să tot fiți întrerupt de mesaje de eroare înainte de a salva înregistrarea. Acest lucru ne conduce la întrebarea fundamentală dacă regulile de validare la nivel de câmp mai sunt de vreun folos în absența unei încercări de salvare a înregistrării. De ce nu să nu verificăm valorile tuturor câmpurilor o singură dată, la salvarea înregistrării? Această întrebare ne conduce la conceptul de reguli la nivel de înregistrare.

Uneori, situația unui câmp depinde de situația unuia sau mai multor câmpuri. În acest context nu se aplică regulile de validare la nivel de câmp. Puteți numai să verificați situația tuturor câmpurilor înainte de a salva înregistrarea.

Adeseori, dezvoltatorii de baze de date se bazează exclusiv pe reguli de validare la nivel de înregistrare. De exemplu, regulile de validare care altfel s-ar aplica la nivel de câmp sunt implementate ca parte a schemei generale de reguli la nivel de înregistrare. Pentru a oferi mai multă claritate, secțiunea următoare va implementa conceptele de validare la nivel de câmp și de înregistrare.

Notă

Poate că vă întrebați cui îi aparține responsabilitatea de a se asigura că în baza de date sunt introduse date valide? În mod tradițional, aceasta este treaba aplicațiilor software. De exemplu, pentru a asigura introducerea adresei complete, care conține adresa, orașul, statul și codul poștal, era utilizat sistemul software de ordonare a intrărilor. Cândva, bazele de date erau numai instrumente de stocare. Probabil că în trecut aceasta nu era o problemă, fiindcă o bază de date era accesată de o singură aplicație. Astăzi însă, bazele de date sunt construite pentru a fi accesibile pentru mai multe aplicații. Ce s-ar întâmpla dacă o aplicație ar omite o regulă de validare importantă? Ultima redută pentru apărarea integrității este chiar baza de date. Dacă vă amintiți cele discutate în capitolul 1 și cele 12 reguli ale lui Codd, regula 5 formulează ideea că o bază de date relațională trebuie să suporte un limbaj care facilitează integritatea datelor.

Doar fiindcă o bază de date respectă regula 5 a lui Codd, nu înseamnă neapărat că trebuie să utilizați aceste funcții ale bazei de date. Dacă, totuși, sunteți preocupat de aplicarea „celor mai bune practici", este recomandabil să determinați regulile esențiale pentru integritatea bazei de date și să implementați aceste reguli. În consecință, indiferent de aplicația utilizată pentru a interacționa cu baza de date, aceasta din urmă „își va purta singură de grijă", iar dumneavoastră veți fi sigur de integritatea datelor.

Implementarea validării la nivel de câmp

Înainte de a intra în detaliile implementării validării la nivel de câmp, trebuie să efectuăm un studiu asupra proprietăților relevante ale unui câmp care suportă validarea. În capitolul 2 am realizat o scurtă prezentare a acestor proprietăți. În acest capitol veți vedea cum funcționarea acestor proprietăți ajută la susținerea integrității datelor.

Proprietăți ale câmpurilor

Figura 6.1 prezintă diferitele proprietăți ale câmpului ClientID din tabelul Client. În funcție de modul în care doriți să configurați regulile, este posibil să utilizați unele dintre aceste proprietăți sau chiar pe toate, deoarece, în funcție de tipul de date ale câmpului, unele proprietăți nu pot fi utilizate. De exemplu, proprietatea Allow Zero Length (permite lungimea zero) poate fi aplicată numai unui câmp cu date de tip text, iar proprietatea Decimal Places (numărul de zecimale) poate fi aplicată numai unui câmp numeric. În cazul câmpurilor Autonumber, proprietatea Required (obligatoriu) nu se aplică, deoarece utilizatorul nu are controlul asupra datelor stocate.

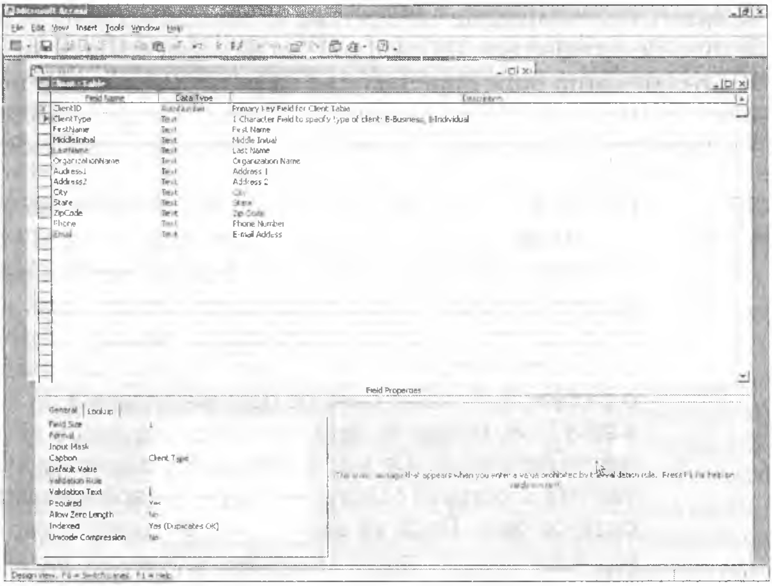


FIGURA 6.1
Access suportă o varietate de proprietăți ale câmpurilor pentru a susține validarea și integritatea datelor.

Această secțiune va prezenta modul de stabilire a regulilor la nivel de câmp pentru tabelul Client din baza de date TEB.

Proprietatea Required

Dacă este obligatorie introducerea unei valori într-un câmp, proprietatea Required este setată pe Yes (da). Figura 6.2 prezintă modul de setare a acestei valori. Această proprietate este de tip boolean, adică poate accepta numai două valori: Yes (da) sau No (nu). Uneori valorile pot fi True (adevărat) / False (fals), sau 1 / 0. O variabilă booleană este ca un comutator – este fie pornit, fie oprit.

> **Indiciu**
> Pentru a inversa valoarea proprietății Required. trebuie doar să selectați valoarea cu mouse-ul și să executați dublu clic. O altă metodă este să executați clic pe săgeata cu vârful în jos, apoi să selectați valoarea dorită din lista derulantă care apare.

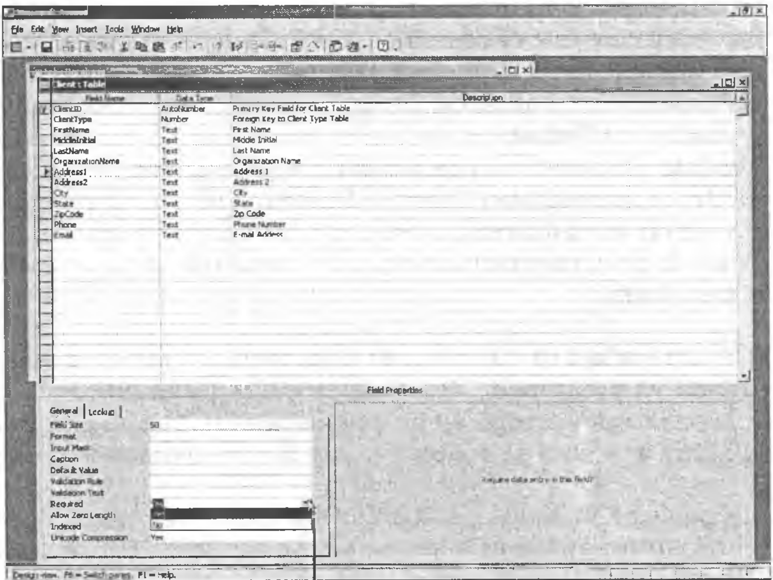
Bunul simț și rezultatele eforturilor dumneavoastră de analiză și proiectare vă vor indica adesea dacă este obligatorie completarea valorii unui câmp. În cazului tabelului Client, completarea câmpurilor `ClientType`, `Address1`, `City`, `State`, `ZIP` și `Phone` este obligatorie. Dar câmpul `Address2`? Nu va fi nevoie de a doua adresă pentru toți clienții. Același lucru este valabil și pentru adresa de `Email`; nu toți clienții au așa ceva. Ce părere aveți despre câmpurile `FirstName`, `MiddleInitial`, `LastName` și `OrganizationName`? Aceste câmpuri sunt o problemă delicată. Unii clienți sunt persoane fizice, alții sunt firme. Dacă un client este persoană fizică, pentru care valoarea câmpului `ClientType` este „I" (de la *individual* – n.tr.), atunci valorile câmpurilor `FirstName` și `LastName` sunt obligatorii.

Completarea câmpului `MiddleInitial` nu este obligatorie, deoarece nu toți oamenii au două nume. Dacă este vorba despre o firmă, atunci valoarea câmpului `ClientType` este „B", deci valoarea câmpului `OrganizationName` este obligatorie. După cum vedeți, în ultimele cazuri, obligativitatea completării unor câmpuri depinde de valorile altor câmpuri. Aici intervin regulile de validare la nivel de înregistrare, iar această situație va fi discutată în secțiunea următoare.

Pentru a recapitula, tabelul 6.1 prezintă setările proprietății Required pentru tabelul Client.

FIGURA 6.2

Proprietatea Required specifică dacă este obligatorie introducerea datelor într-un anumit câmp.



Setările valide pentru proprietatea Required sunt stocate într-o listă derulantă.

Tabelul 6.1 Setările proprietății Required pentru câmpurile din tabelul Client

Câmpul	Valoare
`ClientID`	-
`ClientType`	Yes
`FirstName`	No
`MiddleInitial`	No
`LastName`	No
`OrganizationName`	No
`Address1`	Yes
`Address2`	No
`City`	Yes

Tabelul 6.1 (continuare)

Câmpul	Valoare
`State`	Yes
`ZipCode`	Yes
`Phone`	Yes
`Email`	No

Acum, după ce ați setat corespunzător propietatea Required, este momentul să vedeți care este efectul său asupra datelor introduse. Dacă vă mai amintiți din ultimele două capitole, tabele pot fi vizualizate în două moduri. Primul este modul (vizualizarea) Design, cu care ați lucrat intens. Al doilea este vizualizarea Datasheet, prezentat în capitolul 2, care se comportă ca o foaie de date. În acest mod de vizualizare puteți privi, modifica și șterge datele dintr-un anumit tabel. Figura 6.3 prezintă modul de trecere de la vizualizarea Design la vizualizarea Datasheet. Trebuie doar să executați clic pe butonul View de pe bara principală de instrumente din Access și să selectați vizualizarea dorită.

Când vă aflați în vizualizarea Datasheet, intrați în câmpul `FirstName` și scrieți-vă prenumele. Veți observa că, după ce începeți să scrieți, câmpul `ClientID` este completat cu o nouă valoare. Figura 6.4 prezintă înfățișarea pe care ar trebui să o aibă ecranul dumneavoastră.

După ce ați introdus o valoare în câmpul `FirstName`, încercați să ieșiți din înregistrare apăsând tasta cu săgeata în jos. Când încercați să părăsiți o înregistrare, Access verifică dacă sunt îndeplinite toate regulile de validare stabilite. Dacă regulile sunt îndeplinite, înregistrarea este salvată. Dacă regulile nu sunt îndeplinite, este generat un mesaj de eroare (vedeți figura 6.5), care vă informează că s-a produs o excepție. Așa cum vedeți, nu se poate spune că acesta este un mesaj prietenos față de utilizator. Spre sfârșitul acestei secțiuni veți învăța tehnici suplimentare de a oferi utilizatorilor bazei de date mesaje mai intuitive și mai prietenoase.

În acest moment, dacă executați clic pe butonul OK pentru a scăpa de caseta de dialog și apoi apăsați tasta Escape, noua înregistrare pe care ați dorit să o creați va fi ștearsă. Indiferent cât de mult vă veți strădui, Access nu vă va permite să treceți peste regulile de validare instituite! Să ne concentrăm acum atenția asupra proprietății Allow Zero Length, care este foarte apropiată de proprietatea Required.



FIGURA 6.3
Butonul View vă permite să treceți de la vizualizarea Design la vizualizarea Datasheet și invers.

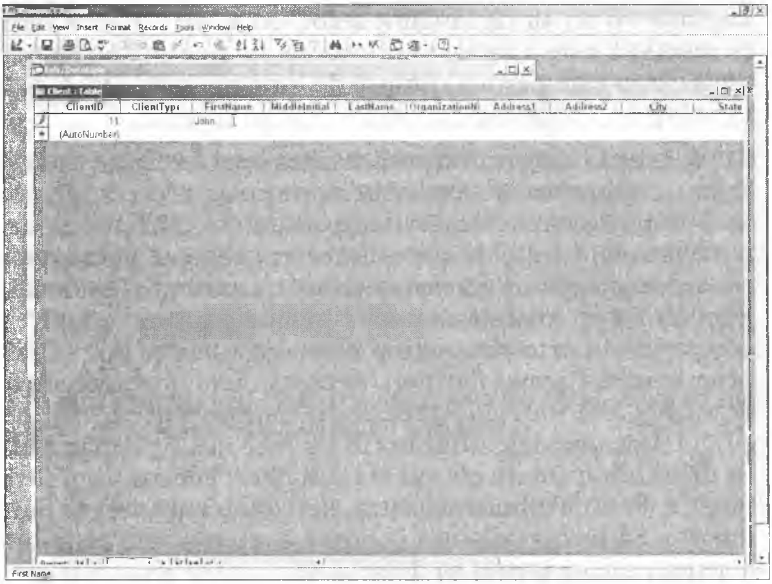


FIGURA 6.4
Imediat ce introduceți date într-o înregistrare nouă, câmpul cu tipul de date `AutoNumber` este completat cu o nouă valoare.

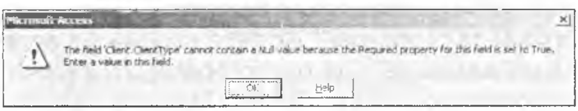


FIGURA 6.5
Nu vi se permite salvarea înregistrării dacă nu sunt îndeplinite regulile de validare.

Proprietatea Allow Zero Length

Această proprietate vi se va părea confuză la început, deoarece este necesar să deosebiți între șirul vid și un șir de lungime zero.

Indiciu

Șirul vid poate fi un concept dificil de înțeles. Gândiți-vă la un câmp liber din înregistrarea unui client. Este el liber cu adevărat sau valoarea sa nu este cunoscută? Șirul vid indică situația când valoarea nu este cunoscută, spre deosebire de cea în care valoarea nu este completată.

Combinația de valori între proprietățile Required și Allow Zero Length influențează funcționarea validării. Cel mai bun punct de start este acela în care proprietatea Required este setată pe Yes. În acest caz, șirul vid nu va fi niciodată acceptat. Nu uitați că el corespunde valorii necunoscute. Faptul că este obligatorie introducerea valorii unui câmp elimină posibilitatea ca ea să fie necunoscută. Atunci, dacă nu este permis șirul vid, cum stau lucrurile cu un șir de lungime zero? Un șir de lungime zero, spre deosebire de unul vid, este o valoare cunoscută, iar în cazul obligativității completării unui câmp este permis câmpul de lungime zero. Dacă este cazul, veți dori să setați pe Yes proprietatea Allow Zero Length.

Să revenim la cazul când proprietatea Required este setată pe No. Un câmp a cărui completare nu este obligatorie poate accepta întotdeauna un șir vid. Se pune întrebarea dacă vi se permite să introduceți un șir de lungime zero. Revenim astfel la necesitatea cunoașterii diferenței între șirul vid și cel de lungime zero. Să considerăm, de exemplu, o adresă de e-mail. Dacă în acest câmp nu este nimic, atunci clientul nu are adresă de e-mail sau adresa lui nu este cunoscută? Vom discuta mai târziu în această secțiune proprietatea Format (format). Aceasta oferă o metodă de a controla modul de afișare a șirului vid. Dacă doriți, este posibil ca șirul vid să apară ca un șir gol. Ceea ce vede utilizatorul nu este nici pe departe la fel de important ca diferența între un șir vid și unul liber. Pentru un utilizator, ele nu sunt diferite. Pentru dumneavoastră, dezvoltatorul bazei de date, ele sunt foarte diferite. Să ne oprim puțin pentru a examina lista câmpurilor, în scopul de a vedea care ar trebui să fie setările.

Câmpul `ClientType` poate fi numai „B" sau „I". Prin definiție, completarea sa este obligatorie, iar cum nu sunt posibile decât două valori, nu trebuie să fie permis un șir de lungime zero. Dar câmpurile `FirstName`, `MiddleInitial`, `LastName` și `OrganizationName`? Ar avea rost să fie acceptat un șir liber ca valoare pentru `LastName`? Cel mai probabil, răspunsul este nu. Aceste câmpuri fie vor conține o valoare, fie nu. Dacă vă gândiți puțin, un șir cu lungimea zero nu va fi acceptabil aproape niciodată. De aceea, în tabelul Client, proprietatea Allow Zero Length va fi setată pe No pentru toate câmpurile. În situația când veți încerca să introduceți un câmp de lungime zero, Access îl va înlocui automat cu un șir vid.

Sfat

Oricât ați încerca, nu veți găsi pe tastatură tasta corespunzătoare șirului vid! Dacă un câmp are proprietatea Allow Zero Length setată pe șirul vid, adică dacă este acceptat șirul nul, trebuie doar să ștergeți toate caracterele din acel câmp. Aceasta va indica programului Access că trebuie să înlocuiască valoarea câmpului cu șirul nul. Imediat ce introduceți un câmp, toate caracterele vor fi evidențiate. Trebuie doar să apăsați bara de spațiu și toate caracterele vor fi șterse.

Proprietatea Format

Așa cum bănuiți, probabil, toate aceste proprietăți sunt legate una de alta într-o anumită măsură. Așa cum proprietatea Allow Zero Length are legătură cu proprietatea Required, proprietatea Format are legătură cu Allow Zero Length. Ea are două roluri. Mai întâi, controlează modul de afișare a câmpurilor, apoi controlează modul de afișare a șirului vid.

Indiferent de modul de stocare a datelor, proprietatea Format poate modifica înfățișarea datelor. De exemplu, dacă toate caracterele câmpului FirstName sunt introduse cu litere mici, proprietatea Format poate forța afișarea caracterelor cu majuscule, fără a afecta modul în care sunt memorate datele. Această proprietate este foarte complexă, cu multe setări care depind de tipul datelor cu care lucrați.

Indiciu

Această carte are două scopuri: mai întâi, să vă învețe cum să proiectați bazele de date, apoi să vă învețe cât se poate de multe despre Access. Documentația ajutătoare pe care o conține acest program este destul de bună, dar informațiile nu au o valoare prea mare. Prezentarea modului de funcționare a aplicațiilor, pe de altă parte, are o valoare incredibilă. După prezentare, puteți trece la experimentarea diferitelor setări. Știind aceasta, dacă doriți să vă extindeți cunoștințele despre o anumită proprietate sau aplicație, trebuie doar să o selectați și să apăsați tasta F1. De exemplu, figura 6.6 prezintă informațiile din Help despre proprietatea Format.

Ironia este că sistemul Help este una dintre funcțiile cele mai valoroase și totuși mai puțin utilizate ale aplicațiilor software.

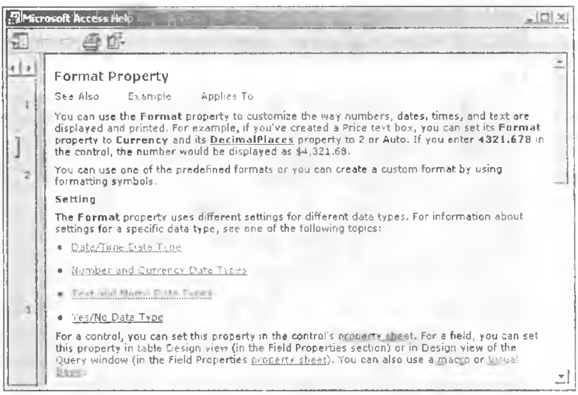
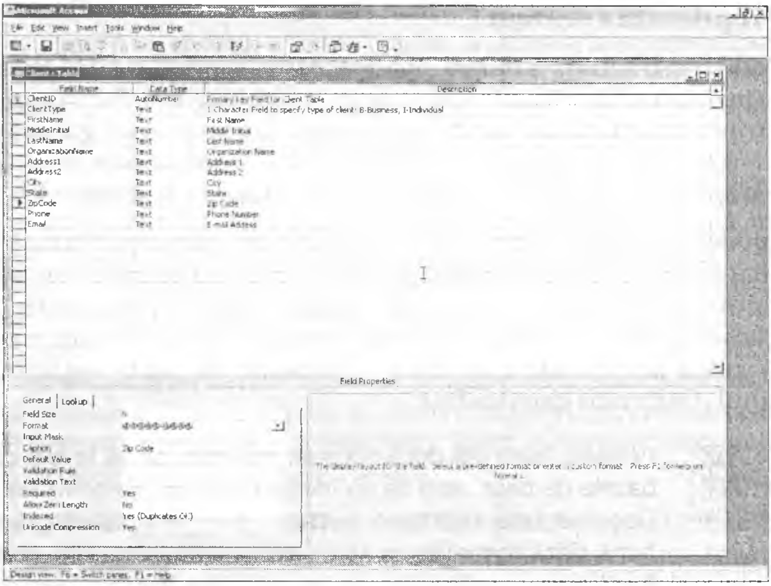


FIGURA 6.6
Dacă apăsați tasta F1 când cursorul este poziționat pe o anumită proprietate va afișa meniul Help pentru acea proprietate.

Proprietatea Format este foarte utilă când trebuie afișate câmpuri cu formatare specială. Două tipuri de astfel de câmpuri sunt cele care conțin numere de telefon și coduri poștale. Figura 6.7 prezintă setarea proprietății Format pentru câmpul ZipCode.

FIGURA 6.7

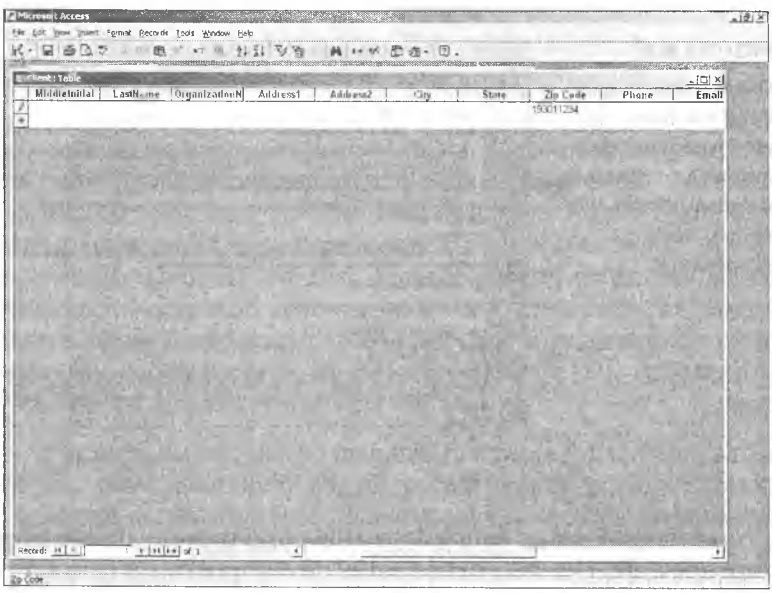
Setarea @@@@@-@@@@ a proprietății Format asigură afișarea codului poștal cu cinci caractere, o liniuță, apoi patru caractere.



Să examinăm acum funcționarea proprietății Format. Reveniți în vederea Datasheet și treceți în `ZipCode`, apoi introduceți 193011234, dar nu treceți încă la câmpul următor. Figura 6.8 prezintă modul în care va arăta vederea Datasheet.

FIGURA 6.8

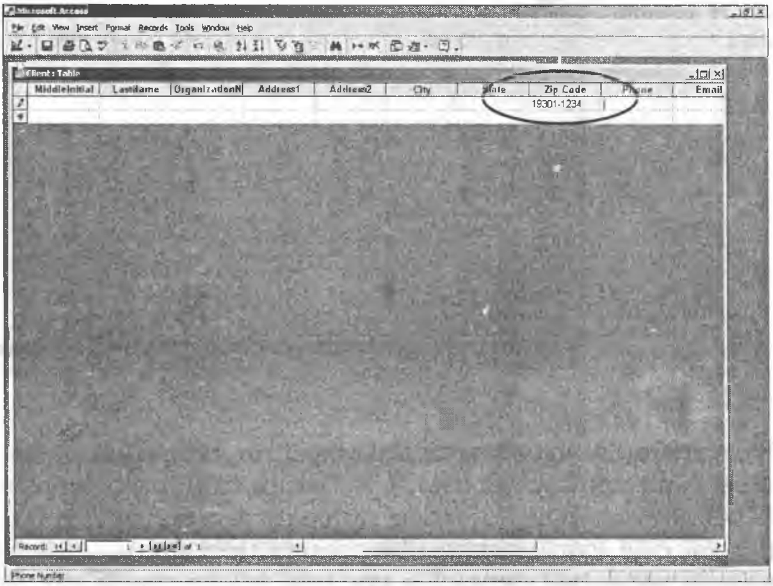
Cât timp vă aflați în câmpul respectiv, proprietatea Format nu are efect.



Treceți acum în câmpul `Phone Number`. Figura 6.9 prezintă efectul proprietății Format. După ce părăsiți un câmp, Access va respecta afișarea stabilită prin proprietatea Format. Nu uitați să apăsați tasta Escape pentru a scăpa de date, astfel încât să puteți reveni la modul Design. Nu uitați, anumite reguli de validare vă vor împiedica să salvați datele.

FIGURA 6.9

Proprietatea Format va avea efect după ce ieșiți din câmp.



> **Notă**
>
> Access nu vă va permite să lăsați datele în stare incertă. De exemplu, dacă în timp ce introduceți date vă decideți să treceți în vizualizarea Design sau să treceți la o nouă înregistrare, va trebui mai întâi să terminați introducerea datelor din înregistrarea curentă. Pentru asta, aveți la dispoziție două opțiuni. Fie veți salva datele, ceea ce înseamnă că trebuie să introduceți valori în câmpuri până sunt îndeplinite condițiile de integritate, fie veți renunța la date. O variantă de mijloc nu există în această situație. Acest concept va fi important la crearea formularelor pentru introducerea datelor, pentru a întreține datele. Dacă utilizatorul se află într-o sesiune de introducere a datelor și decide să închidă formularul, va trebui să aleagă între introducerea completă și renunțarea la date. Aceasta este esența integrității datelor, faptul că nu pot fi stocate date incomplete.

Să ne îndreptăm acum atenția către a doua funcție a proprietății Format, controlul afișarii șirului vid. În cazul câmpurilor de text, proprietatea Format este alcătuită din două secțiuni. Prima se ocupă de modul de afișare a datelor. A doua indică modul de afișare a șirului vid. Pentru baza de date TEB, va fi folosită valoarea „N/A" pentru afișarea șirului vid. Caracterul punct și virgulă (;) separă secțiunile proprietății Format (vedeți figura 6.10).

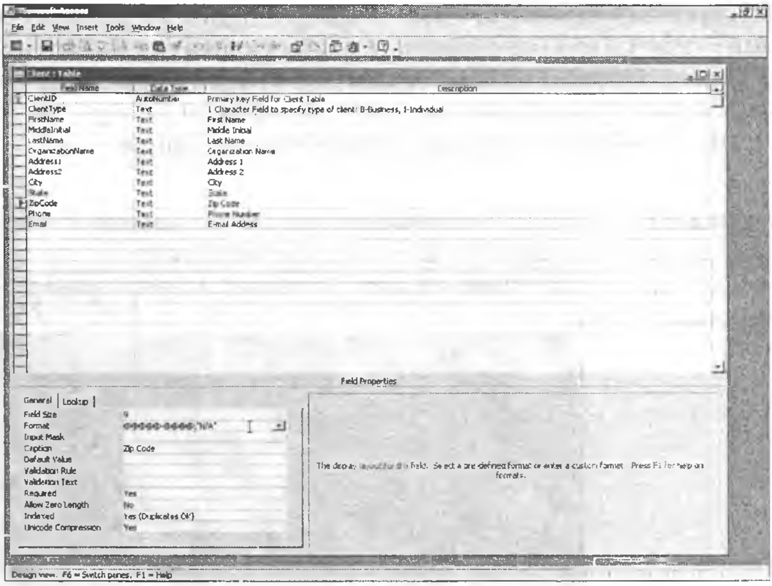


FIGURA 6.10 A doua secțiune a proprietății Format pentru un câmp de text controlează modul de afișare a șirului vid.

Să revenim la vizualizarea Datasheet, în câmpul `ZipCode`. Observați că acest câmp conține valoarea N/A. Dar nu vă grăbiți! Completarea câmpului `ZipCode` este obligatorie, deci nu poate accepta șirul vid. Și atunci? Răspunsul este foarte simplu. La crearea unei noi înregistrări, cu excepția cazului când este definită o anumită valoare implicită, iar despre acest concept vom discuta mai târziu în această secțiune, valoarea implicită a unui câmp este șirul vid. Câmpul așteaptă literalmente să introduceți o valoare inițială. Deoarece completarea câmpului `ZipCode` este obligatorie, nu veți putea salva înregistrarea dacă acest câmp conține șirul vid. Figura 6.11 arată cum va apărea inițial câmpul `ZipCode`.

Este proprietatea Format suficientă pentru a asigura integritatea datelor? Răspunsul este, categoric, nu. Figura 6.12 ilustrează acest lucru. Nu uitați, proprietatea Format controlează modul de afișare a datelor, nu modul în care sunt introduse. Pentru a controla acest aspect, va trebui să studiați proprietatea Input Mask (mască de intrare).

În practică, nu va trebui să controlați modul de afișare a șirului vid în câmpul `ZipCode`, deoarece completarea acestuia este obligatorie. Nu uitați, șirul vid nu poate fi stocat în câmpurile a căror completare este obligatorie. Imaginea anterioară care utiliza câmpul `ZipCode` a avut numai scop demonstrativ. Tabelul 6.2 conține setările pentru fiecare câmp al tabelului Client care necesită o setare pentru proprietatea Format.

FIGURA 6.11

În absența unei valori implicite, câmpurile unei noi înregistrări au valoarea inițială nulă.

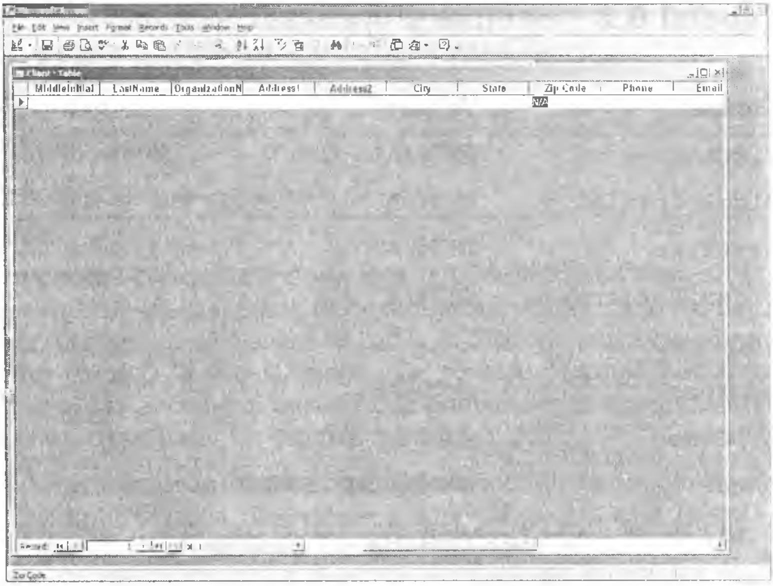
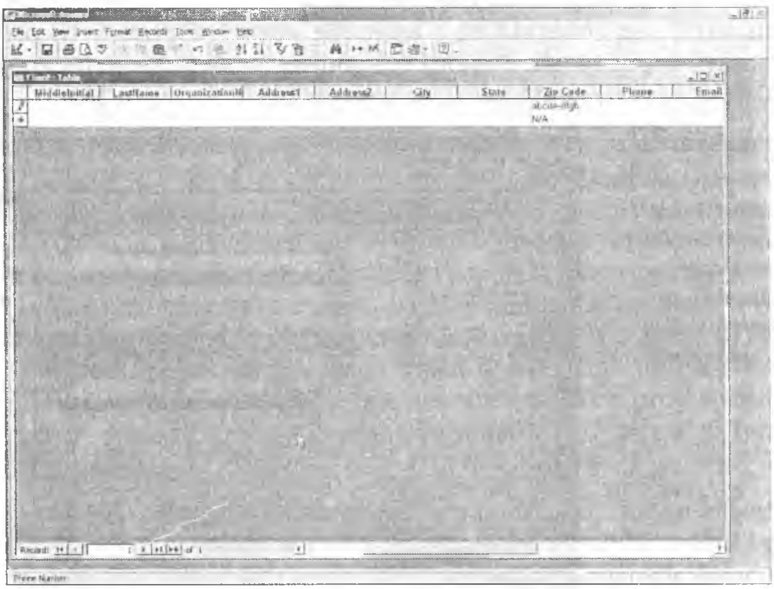


FIGURA 6.12

Proprietatea Format nu poate împiedica introducerea unor caractere nevalide într-un câmp.



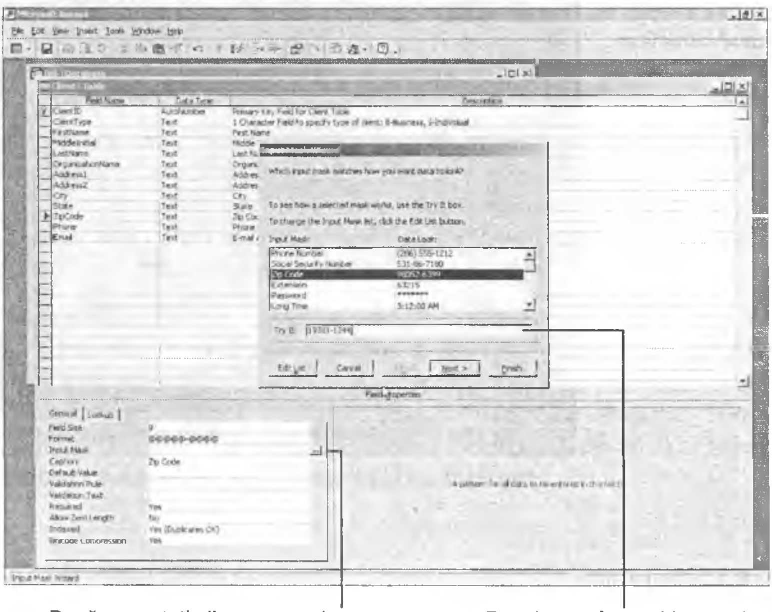
Tabelul 6.2 Setările proprietății Format pentru câmpurile tabelului Client

Câmpul	Valoare
FirstName	@;"N/A"
MiddleInitial	@;"N/A"
LastName	@;"N/A"
OrganizationName	@;"N/A"
Address1	-
Address2	@;"N/A"
ZipCode	@@@@@-@@@@
Phone	@@@-@@@-@@@@
Email	@;"N/A"

Proprietatea Input Mask

Proprietatea Input Mask (mască de intrare) realizează ceea ce proprietatea Format nu poate. Problema este de a controla ce caractere sunt introduse într-un câmp și în ce mod. Proprietatea Input Mask poate fi configurată manual sau generată cu o aplicație wizard. Figura 6.13 prezintă aplicația wizard Input Mask. Tabelul 6.3 conține setările pentru câmpurile tabelului Client care necesită configurarea proprietății Input Mask.

Figura 6.13 Aplicația wizard Input Mask generează rapid o valoare pentru proprietatea Input Mask.



Dacă executați clic pe acest buton, va fi lansată aplicația wizard Input Mask.

Puteți testa în mod interactiv proprietatea Input Mask.

Tabelul 6.3 Setările proprietății Input Mask pentru câmpurile tabelului Client

Câmpul	Valoare
ZipCode	00000-9999;;_
Phone	(999) 000-0000;;_

Masca de intrare pentru câmpul `ZipCode` este 00000-9999. Ea reprezintă un model pe care trebuie să-l respecte toate datele introduse. În acest caz, primele cinci caractere trebuie să fie cifre de la 0 la 9. Ultimele patru caractere trebuie să fie cifre de la 0 la 9 sau spațiu. De ce este permisă folosirea spațiului în partea plus 4 a codului poștal? Dar poate nu știți ce reprezintă partea plus 4 a codului poștal.

Ca și proprietatea Format, proprietatea Input Mask este alcătuită din două părți. În acest exemplu, masca de intrare a codului poștal este 00000-9999;;_. Prima secțiune este chiar masca de intrare. A doua secțiune specifică dacă este prezentată afișarea actuală, inclusiv caracterele modelului, sau dacă sunt stocate datele esențiale. În acest caz, dacă a doua secțiune ar fi fost setată pe 0, la introducerea datelor 19301-1344, Access ar fi încercat să memoreze 10 caractere în câmpul `ZipCode`. În acest caz, al zecelea caracter ar fi liniuța care separă cele două părți ale codului poștal. Dacă lăsați necompletată a doua secțiune, nu vor fi memorate caractere ale modelului, iar în acest caz, datele momorate în câmpul `ZipCode` sunt „193011344". Setarea acestei secțiuni este foarte importantă, deoarece, în acest exemplu câmpul `ZipCode` are lungimea de 9 caractere, iar dacă ar fi memorate caracterele modelului, codul poștal ar fi trunchiat, ceea ce înseamnă că dacă introduceți 19301-1344, va fi memorat de fapt 19301-134. Ultima secțiune specifică înlocuitorul (*placeholder*) pentru spațiu la introducerea datelor. În acest caz este folosit caracterul underscore (bara jos), dar el poate fi orice caracter înlocuitor. Aplicația wizard Input Mask cuprinde trei pași. Fiecare pas corespunde unei secțiuni. În majoritatea situațiilor, veți lăsa neschimbate setările implicite pe care vi le oferă aplicația. Figura 6.14 prezintă modul cum va apărea proprietatea Input Mask după închiderea aplicației wizard. Pe măsură ce veți înțelege mai bine funcționarea proprietății Input Mask, veți introduce chiar dumneavoastră măștile de intrare, în loc de a folosi aplicația wizard.

Acum, după ce aveți una dintre măștile de intrare la degetul mic, mergeți mai departe și setați câmpul `Phone Number`. Figura 6.15 prezintă modul în care vor apărea rezultatele. Observați semnul exclamării care precede masca de intrare pentru câmpul `Phone Number`? Acest lucru determină afișarea caracterelor de la dreapta la stânga.

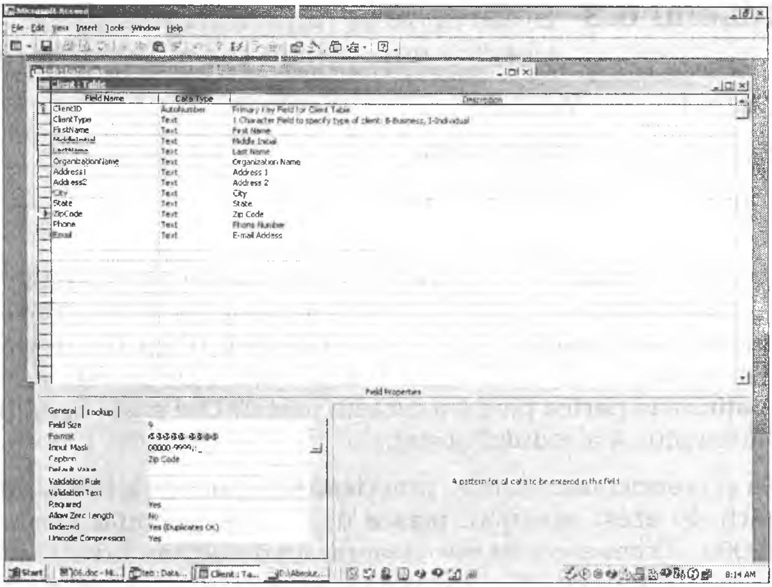


FIGURA 6.14
Aveți de ales între utilizarea unei aplicații wizard și setarea manuală a proprietății.

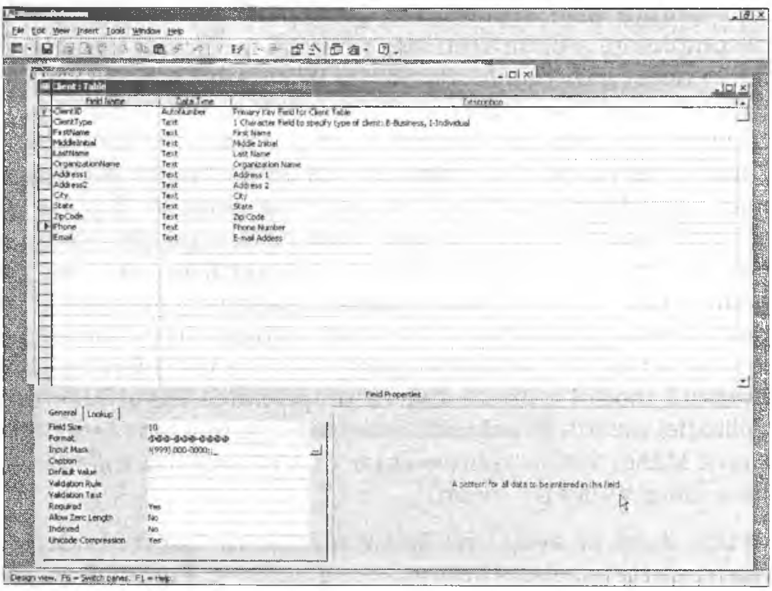


FIGURA 6.15
Caracterul ! determină afișarea caracterelor de la dreapta la stânga.

Indiciu

Când salvați tabelul, veți observa probabil că s-a schimbat proprietatea Input Mask pentru câmpurile ZipCode și Phone. Masca de intrare pentru câmpul ZipCode s-a transformat în 00000\-9999;;_, iar pentru Phone s-a transformat în !\(999") „000\-0000;;_. Caracterul \ face să fie afișate caracterele de după el ca și caractere literale. În majoritatea situațiilor, nu trebuie să vă faceți probleme privind modul de stocare a acestei informații în Access.

Să ne oprim câteva momente pentru a efectua o testare, revenind la vederea Datasheet. Intrați în câmpul `ZipCode` și scrieți 19301, apoi treceți în câmpul `Phone`. Figura 6.16 prezintă problema care apare.

Soluția acestei probleme este foarte simplă. Caracterul ! funcționează oarecum altfel decât pentru proprietatea Format. Când la începutul proprietății Format apare caracterul !, datele sunt aliniate la stânga. De aceea, dacă modificați proprietatea Format în !@@@@@-@@@@, datele vor fi afișate corespunzător când nu este introdusă partea plus 4 a codului poștal (vedeți figura 6.17).

FIGURA 6.16

Este important să vă asigurați că ați corelat proprietățile Format și Input Mask.

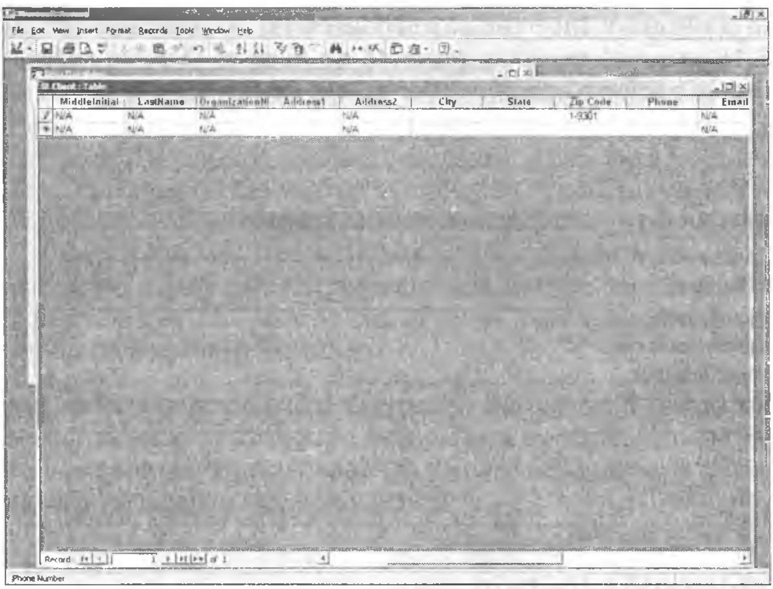
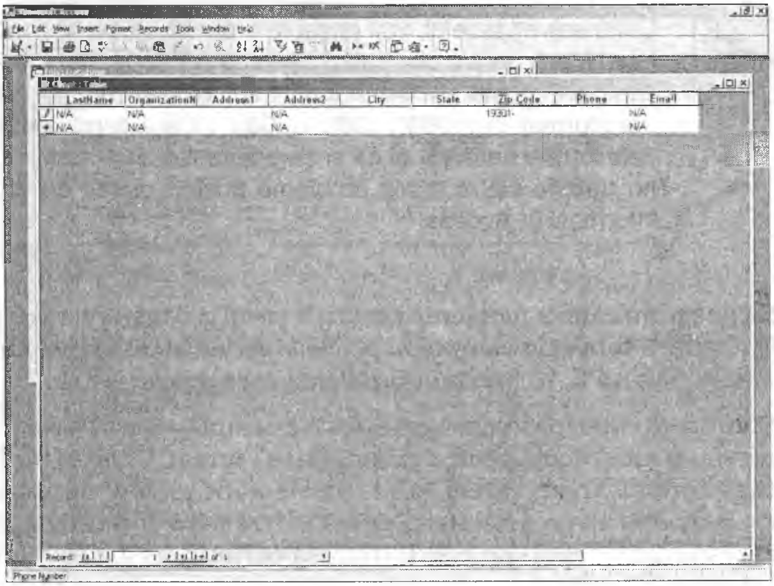


FIGURA 6.17

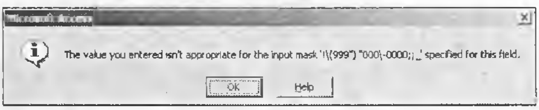
Caracterul ! din proprietatea Format determină afișarea caracterelor aliniate la stânga.



În continuare, încercați să introduceți un număr de telefon incomplet. Figura 6.18 prezintă excepția generată de Access. Vi se va solicita să introduceți datele conform specificațiilor măștii de intrare.

FIGURA 6.18

Access vă solicită să introduceți datele conform specificațiilor măștii de intrare.



Proprietatea Default Value

Proprietatea Default Value (valoarea implicită) face ceea ce indică și numele ei – specifică valoarea care este introdusă automat într-un câmp la crearea unei noi înregistrări. Pentru câmpul `ClientType` este specificată valoarea implicită I (de la *Individual* = persoană fizică), pentru că s-a ajuns la concluzia că majoritatea clienților sunt persoane fizice. Figura 6.19 prezintă setarea proprietății Default Value pentru câmpul `ClientType`. Acesta este singurul pentru care este setată o valoare implicită.

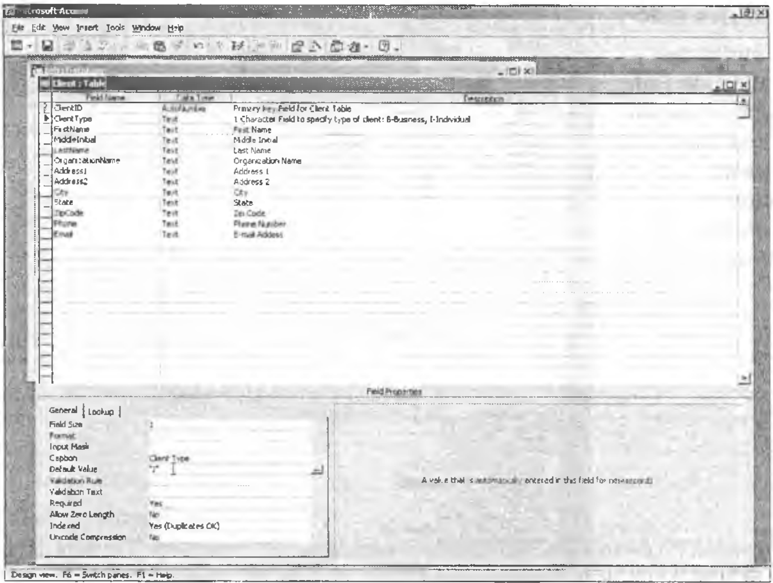


FIGURA 6.19 Proprietatea Default Value vă permite să specificați o valoare inițială a câmpului când este creată o nouă înregistrare.

Proprietatea Validation Rule

Subiectul discuției se îndreaptă acum spre una dintre cele mai importante funcții din Access și din orice produs pentru baze de date în general – capacitatea de a specifica reguli de validare. Regulile pot fi formulate la nivel de câmp sau de înregistrare. Primul tip de reguli pe care îl vom prezenta este cel la nivel de câmp. Dacă vă amintiți, câmpul `ClientType` poate avea numai două valori: I sau B. în acest moment, nu există o cale de a impune această regulă. Deși este specificată valoarea implicită I, nimic nu vă poate opri să introduceți orice valoare în acest câmp. Dacă vă deplasați la proprietatea Validation Rule, veți vedea un mic buton. Acesta lansează o aplicație de construire a expresiilor (*Expression Builder*), pe care o puteți utiliza pentru a formula regula. Alternativa ar fi să introduceți manual regula, evitând aplicația Expression Builder.

> **Notă** Expresiile pe care le poate utiliza Expression Builder pot fi complexe. Acest capitol tratează numai regulile simple. Dacă nu sunteți familiarizat cu logica booleană, este recomandabil să vă opriți puțin pentru a studia documentația Help pentru reguli de validare.

Figura 6.20 prezintă setarea proprietății Validation Rule pentru câmpul ClientType. Logica este simplă – valoarea acestuia poate fi B sau I.

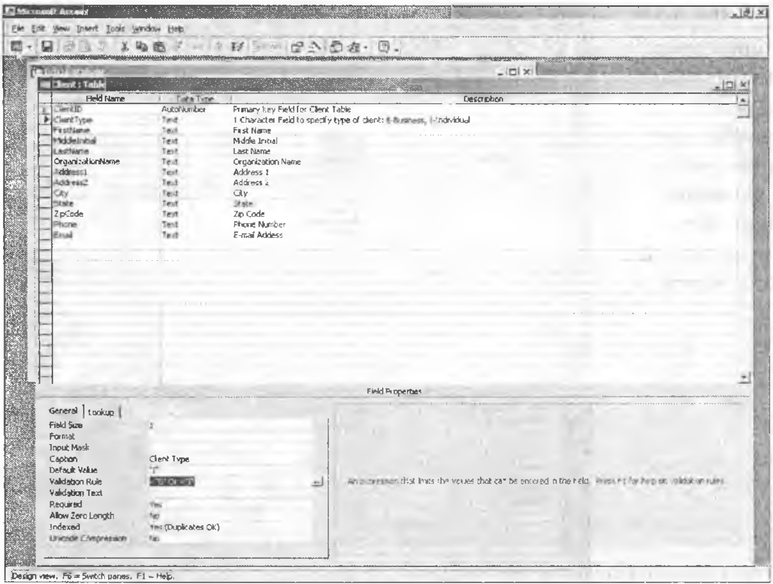


FIGURA 6.20
Proprietatea Validation Rule pentru câmpul ClientType solicită ca valoarea introdusă să fie B sau I.

Să ne oprim puțin pentru a testa regula, revenind în vizualizarea Datasheet. Încercați să introduceți o valoare diferită de B sau I și să părăsiți câmpul. Figura 6.21 prezintă excepția generată de Access.

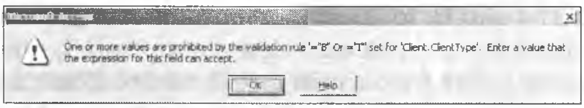


FIGURA 6.21
Când este încălcată o regulă de validare, Access generează o excepție și apare o casetă de dialog care informează utilizatorul despre acest lucru.

Este puțin spus că acest mesaj nu este prietenos față de utilizator. Pentru a rezolva această problemă, să revenim la proprietatea Validation Text (text de validare). Tabelul 6.4 prezintă regulile la nivel pentru fiecare câmp care trebuie să fie setat.

Tabelul 6.4 Setările proprietății Validation Rule pentru câmpurile tabelului Client

Câmpul	Valoarea
`ClientType`	="B" or „I"
`Address1`	Is not Null
`City`	Is not Null
`State`	Is not Null
`ZipCode`	Is not Null
`Phone`	Is not Null

Proprietatea Validation Text

Proprietatea Validation Text (text de validare) este foarte simplă. Trebuie doar să precizați textul care va fi afișat când este încălcată o regulă de validare (vedeți figura 6.22).

Acum reveniți la vederea Datasheet și introduceți o valoare nevalidă în câmpul `ClientType`. Figura 6.23 prezintă mesajul de eroare direct care apare.

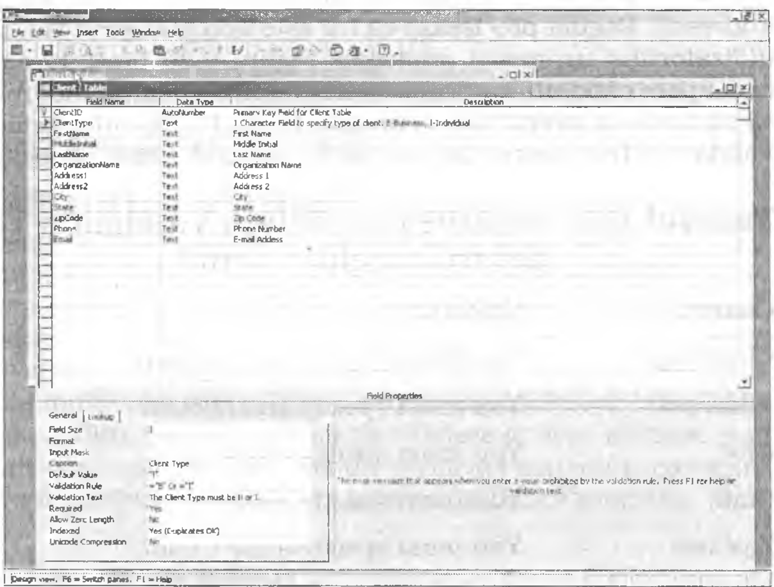


FIGURA 6.22 Proprietatea Validation Text precizează textul care trebuie să fie afișat când un utilizator încalcă o regulă de validare.

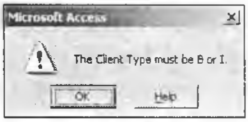


FIGURA 6.23
După ce specificați textul de validare, Access îl va folosi în caseta de validare care informează despre încălcarea unei reguli de validare.

Înainte de a continua, să dăm un alt exemplu de regulă de validare / text de validare. Figura 6.24 prezintă mesajul de eroare pe care îl va vedea utilizatorul dacă lasă câmpul `Address1` necompletat. Din nou, mesajul afișat nu este deloc prietenos.

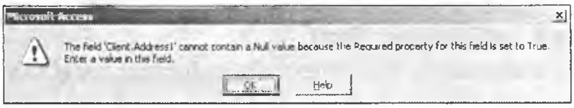


FIGURA 6.24
Acesta este mesajul de eroare implicit generat de Access când lăsați necompletat un câmp a cărui valoare este necesară.

Figura 6.25 prezintă regula de validare ș textul de validare pentru câmpul Address1. Regula precizează că nu este acceptat șirul vid, iar textul oferă utilizatorului un mesaj prietenos. Figura 6.26 prezintă caseta de dialog care apare când utilizatorul lasă necompletat câmpul Address1 și încearcă să părăsească înregistrarea curentă. Tabelul 6.5 conține setările proprietății Validation Text pentru fiecare dintre câmpurile care necesită acest lucru.

Tabelul 6.5 Setările proprietății Validation Text pentru tabelul Client

Câmpul	Valoarea
ClientType	The Client Type must be B or I.
Address1	You must specify an address.
City	You must specify a city.
State	You must specify a state.
ZipCode	You must specify a ZIP Code.
Phone	You must specify a phone number.

(Valoarea proprietății Validation Text, adică mesajul de eroare pe care îl va vedea utilizatorul la încălcarea unei reguli de validare, nu trebuie neapărat să fie în limba engleză; de altfel, un text în limba română ar fi mai prietenos pentru utilizatorul român – n. tr.)

FIGURA 6.25

Regula de validare pentru câmpul Address1 precizează că acest câmp nu poate fi lăsat necompletat.

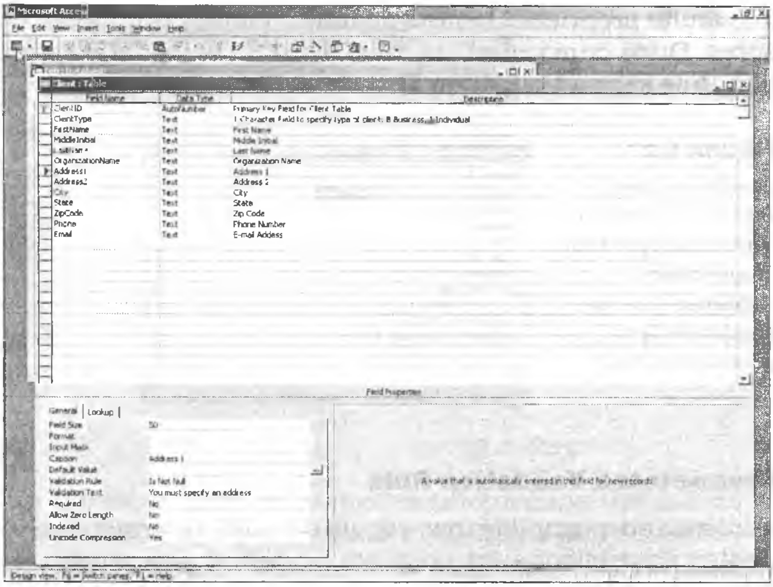


FIGURA 6.26

Proprietatea Validation Text oferă un text prietenos față de utilizator.



Implementarea validării la nivel de înregistrare

Când încercați să salvați o înregistrare, Access verifică mai întâi dacă sunt satisfăcute regulile de validare. Ați văzut acest lucru și în exemplele anterioare. Dacă sunt satisfăcute regulile de validare la nivel de câmp, Access verifică îndeplinirea regulilor la nivel de înregistrare, uneori denumite reguli de validare la nivel de tabel. Regulile la nivel de tabel sunt utile când trebuie să comparați mai multe câmpuri pentru a determina dacă sunt îndeplinite sau nu anumite condiții. Logica regulilor de validare la nivel de tabel este adesea mult mai complexă, deoarece trebuie verificate mai multe câmpuri. Acest concept va fi mai bine ilustrat prin intermediul unui exemplu.

Proprietăți la nivel de câmp

Dacă vă amintiți, trebuie specificate numele și prenumele clientului care este persoană fizică, respectiv numele firmei. Figura 6.27 prezintă, la care aveți acces dacă apăsați butonul Properties, de pe bara de instrumente Main Access. Caseta de dialog Table Properties vă permite să specificați

mai multe proprietăți la nivel de tabel, inclusiv proprietăți înrudite cu validarea. După cum vedeți, un tabel, ca și un câmp, are o proprietate Validation Rule și una Validation Text.

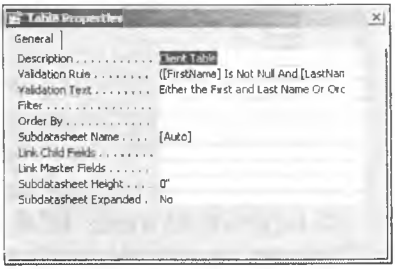


FIGURA 6.27
Caseta de dialog Table Properties vă permite să specificați proprietățile Validation Rule și Validation Text pentru un tabel.

Proprietatea Validation Rule

Continuând exemplificarea, regula de validare următoare vă va asigura că pentru orice client sunt precizate numele și prenumele, respectiv numele firmei, după caz:

```
([FirstName] Is Not Null And [LastName] Is Not Null) Or
([OrganizationName] Is Not Null)
```

Mergeți mai departe și specificați valorile pentru `Address1`, `City`, `State`, `ZipCode` și `Phone`, lăsând necompletate câmpurile `FirstName`, `LastName` și `OrganizationName`. Figura 6.28 prezintă caseta de dialog cu mesajul de eroare pe care îl va vedea utilizatorul. Veți utiliza proprietatea Validation Text pentru a rezolva această problemă.

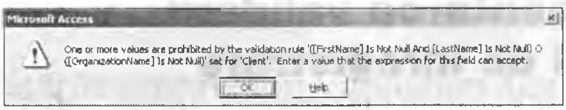


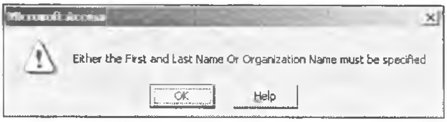
FIGURA 6.28
După îndeplinirea tuturor regulilor la nivel de câmp, Access va testa regula la nivel de rând.

Proprietatea Validation Text

Așa cum ați văzut deja, proprietatea Validation Text vă prmite să creați mesaje prietenoase față de utilizator. În acest caz configurați proprietatea Validation Text astfel: trebuie specificate fie numele și prenumele clientului (dacă este persoană fizică), fie numele firmei. Mergeți mai departe și repetați testul anterior. Figura 6.29 prezintă mesajul prietenos față de utilizator care apare (nu uitați, mesajul depinde de textul introdus, care nu trebuie neapărat să fie în limba engleză – n.tr.).

FIGURA 6.29

Proprietatea Validation Text pentru un tabel oferă un mesaj prietenos față de utilizator.



După ce introduceți numele și prenumele sau numele firmei, iar celelalte reguli sunt satisfăcute, veți putea să treceți la o nouă înregistrare, salvând înregistrarea curentă.

Rezumat

Să nu credeți că o bază de date va împiedica întotdeauna utilizatorul să introducă date nevalide. Pentru a menține integritatea bazei de date, fie aplicația pentru baze de date, fie chiar baza de date trebuie să fie dotate cu logica necesară pentru validarea datelor. Dacă introduceți logica de validare în baza de date, veți fi întotdeauna asigurat privind integritatea datelor, deoarece, la urma urmei, toate aplicațiile trebuie să utilizeze baza de date pentru a salva datele. Așa cum ați văzut, există două niveluri de validare: la nivel de câmp și la nivel de înregistrare (sau rând). Validarea la nivel de înregistrare mai este denumită validare la nivel de tabel. Când salvați datele, baza de date va verifica mai întâi dacă sunt îndeplinite regulile la nivel de câmp. Dacă ele sunt îndeplinite, vor fi verificate apoi regulile la nivel de înregistrare. Dacă și acestea sunt îndeplinite, sunteți la un pas de a putea salva datele. După cum veți vedea în capitolul următor, încă nu ați scăpat de probleme. Conceptul de integritate a datelor se extinde la relațiile dintre entități. Acest lucru este numit *integritate referențială* și ne vom ocupa de el în capitolul următor.



7

BAZELE INTEGRITĂȚII REFERENȚIALE

În acest capitol:

- Ce este integritatea referențială?
- Importanța menținerii integrității referențale
- Funcționarea integrității referențiale
- Rezumat

Capitolul anterior v-a prezentat conceptul de validare a datelor. Recapitulând, validarea datelor, fie la nivel de câmp sau de înregistrare, are ca scop să se asigure că în baza de date sunt salvate numai date valide. Validarea datelor se ocupă de tabelul în care sunt definite regulile. Cu alte cuvinte, regulile de validare pentru tabelul Client au ca obiect impactul noilor date asupra acestui tabel. Impactul noilor date despre clienți asupra tabelului Contact depășește scopul regulilor de validare. În timp ce regulile de validare se ocupă de problemele din interiorul unui tabel, regulile de integritate referențială (*referential integrity – RI*) se ocupă de problemele dintre tabele. De exemplu, este permisă ștergerea unei anumite înregistrări? Dacă da, ce se va întâmpla cu datele din tabele înrudite? Este permisă crearea de date într-un tabel copil? Nu se poate răspunde la aceste întrebări în contextul unui singur tabel. Problemele pe care le ridică aceste întrebări și răspunsurile lor sunt discutate în acest capitol. Anexa B conține o listă detaliată a regulilor de integritate referențială pentru baza de date Time Entry and Billing.

Ce este integritatea referențială?

Integritatea referențială (*referential integrity – RI*) este un sistem care asigură menținerea validității relațiilor dintre tabele. Gândiți-vă la exemplul clasic al comenzilor și al detaliilor despre acestea. Folosind ceea ce ați învățat până acum, știți deja că aceasta este o relație unu la mulți și că această relație este facilitată de preluarea cheii primare din comenzi în detalii despre comenzi. Concluzia logică a acestui lucru este că, dacă nu există o comandă, nu poate exista un detaliu despre aceasta. Se pune întrebarea „Cum pot fi impuse regulile dintre tabele?" Răspunsul constă în crearea a ceea ce se numește *restricții pentru integritatea referențială (RI constraints)*. Deși bazele de date ca cele din Access au capacitatea de a suporta integritatea referențială, trebuie să echipați baza de date cu știința de a utiliza această capacitate.

Importanța menținerii integrității referențiale

Menținerea integrității referențiale este importantă, din aceleași motive pe care le-am prezentat în capitolul anterior, la validarea datelor. Capacitatea unei baze de date de a oferi informații corecte este direct legată de calitatea datelor. Capacitatea de a oferi informații corecte este direct legată și de calitatea relațiilor dintre tabele. Importanța menținerii integrității referențiale va fi foarte clară în capitolul următor, când vom prezenta limbajul SQL (Structured Query Language). Până atunci, secțiunea următoare va servi ca scurtă prezentare pentru ce urmează.

Să considerăm exemplul ipotetic simplu al clienților și tipurilor acestora. Un anumit tip de client poate caracteriza unul sau mai clienți. De aceea, cheia primară a tipului de client este preluată din tabelul cu date despre clienți. Este vorba despre o relație simplă de tip unu la mulți și ar trebui ca până acum să fiți familiarizat cu acest concept. Continuând exemplul ipotetic, să presupunem că doriți să creați un extras cu date grupate după tipul de client. Cu alte cuvinte, extrasul va conține descrierea tipului de client și datele selectate despre acesta. Pentru a realiza acest extras, aveți nevoie de informații și din tabelul despre clienți și din tabelul cu date despre tipul clienților. Așa cum veți vedea în capitolul următor, care tratează limbajul SQL, pentru a realiza acest lucru trebuie să aplicați o tehnică denumită realizarea unei legături sau a unei uniuni (*join*) între cele două tabele. Întrebarea pe care o veți pune este „Cum pot realiza o legătură între două tabele?". Răspunsul este reprezentat de relația dintre tabele. Nu uitați că tabelele conțin o informație comună: identificatorul tipului de client. Aceasta este legătura cerută pentru uniune, iar uniunea vă va permite să combinați datele din tabele diferite. Ce s-ar întâmpla dacă ar fi permisă ruperea legăturii? Să luăm în considerare următoarele întrebări:

- Este posibil să atribuiți unui client un tip inexistent?
- Este posibil să ștergeți un tip de client care a fost atribuit cel puțin unui client?

Aceste întrebări au un element comun: o legătură ruptă. Consecința ruperii legăturii este pierderea datelor. Mai precis, orice client căruia i se atribuie un tip inexistent va fi omis din extrasul de date. Să ne gândim ce-ar fi dacă vi s-ar permite să ștergeți un tip de client care a fost atribuit deja cel puțin unui client. Dacă ștergerile nu sunt restricționate, riscați să compromiteți integritatea relației. Restricțiile privind integritatea referențială vă apără exact de acest tip de situații. Cel mai bun indicator al integrității bazei de date este calitatea și acuratețea informațiilor pe care le produce. Conceptele prezentate în această secțiune vor deveni mai clare în capitolul următor, când veți studia limbajul SQL.

Funcționarea integrității referențiale

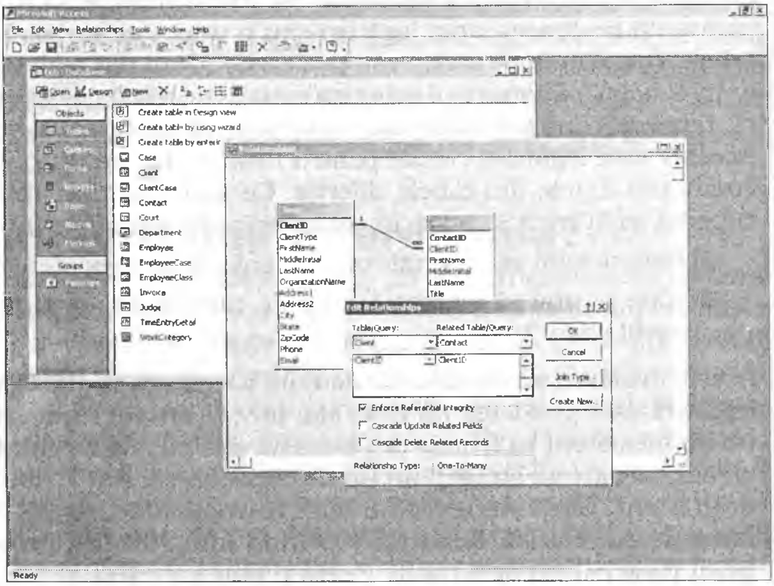
Există patru tipuri de restricții privind integritatea referențială. Acestea sunt: ștergerile restricționate, introducerea restricționată a datelor, ștergerile în cascadă și actualizările în cascadă. Numele fiecărui tip de restricții descrie cu acuratețe funcția respectivă. Secțiunea următoare prezintă aceste patru tipuri de restricții privind integritatea referențială, precum și modul în care pot fi aplicate. Fiecare tip va fi exemplificat în contextul tabelelor Client și Contact din baza de date Time Entry and Billing.

Stabilirea regulilor de integritate referențială

Înainte de a vedea cum funcționează restricțiile, baza de date trebuie să fie echipată cu capacitatea de a suporta integritatea referențială. În capitolul 5 ați creat relații între tabelele din baza de date Time Entry and Billing. Definirea relațiilor dintre aceste tabele este primul pas pentru a da bazei de date capacitatea de a impune integritatea referențială. Apare caseta de dialog Edit Relationships, care vă permite să specificați modul în care vor fi impuse restricțiile privind integritatea referențială (vedeți figura 7.1).

FIGURA 7.1

Caseta de dialog Edit Relationships vă permite să specificați modul în care sunt impuse restricțiile privind integritatea referențială.



Dacă doriți să impuneți integritatea referențială, va trebui să activați (să bifați) opțiunea Enforce Referential Integrity. Figura 7.1 prezintă caseta de validare prin care puteți activa această opțiune. După activarea ei, mai puteți activa, opțional, două tipuri de restricții. Acestea sunt actualizările în cascadă și ștergerile în cascadă. Ce implicație va avea acest lucru asupra celorlalte două tipuri de restricții? Singurul tip care nu este opțional este cel al ștergerilor restricționate. Nu va avea sens vreodată să permiteți existența unei înregistrări copil care referă la o înregistrare părinte inexistentă.

Ștergerile restricționate pot fi considerate opusul ștergerilor în cascadă. Înregistrările părinte se pot afla în una sau două situații. Mai întâi, dacă încercați să ștergeți o înregistrare părinte, atunci va trebui să ștergeți și înregistrările copil corespunzătoare. Altfel, veți fi în situația de a avea înregistrări copil care referă înregistrări părinte inexistente. În al doilea rând, dacă încercați să ștergeți o înregistrare părinte care are înregistrări

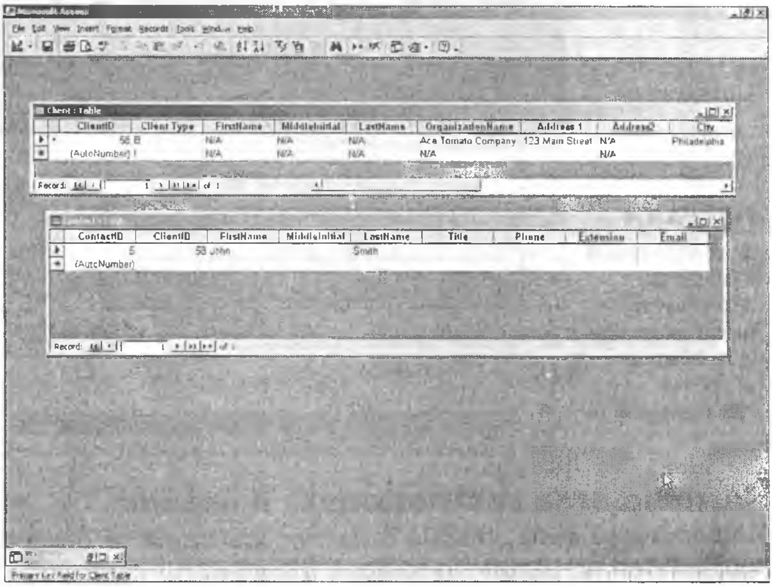
copil, nu vi se va permite acest lucru. Dacă doriți să ștergeți înregistrarea părinte, va trebui să ștergeți mai întâi înregistrările copil. Așa cum puteți vedea, ștergerile restricționate și ștergerile în cascadă se exclud reciproc.

Ștergerile restricționate

Acest tip de restricții înseamnă că o înregistrare părinte nu poate fi ștearsă dacă există una sau mai multe înregistrări copil. Pentru a suporta ștergerile restricționate, caseta de validare Cascade Delete Related Records din caseta de dialog Edit Relationships trebuie să fie nu fie bifată. Figura 7.2 prezintă înregistrările din tabelele Client și Contact care sunt utilizate pentru a demonstra funcționarea integrității referențiale. Figura 7.3 prezintă capacitatea programului Access de a prezenta (afișa) simultan înregistrarea părinte și înregistrările copil.

FIGURA 7.2

Tabelele Client și Contact în vizualizarea Datasheet.



Dacă vă amintiți, în vizualizarea Datasheet puteți efectua orice operație cu date. Pentru a șterge o înregistrare, trebuie doar să selectați întregul rând, executând clic pe marcajul înregistrării din extremitatea stângă a vizualizării Datasheet. Întrucât sunt activate ștergerile restricționate, când încercați să ștergeți o înregistrare din tabelul Client înrudită cu una din tabelul Contact, Access generează o casetă de dialog cu un mesaj de eroare, prezentată în figura 7.4.

Dacă executați clic pe +/-, puteți expanda/contracta afișarea datelor copil.

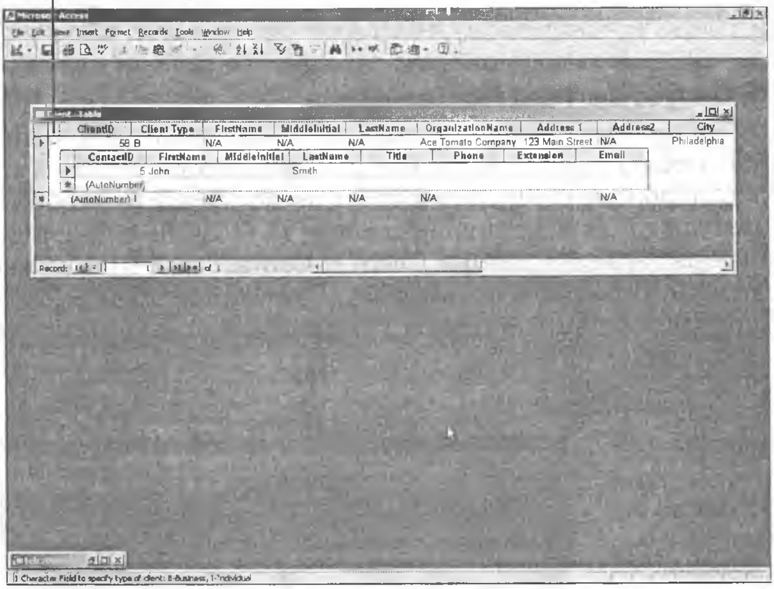


FIGURA 7.3

Access are capacitatea de a afișa simultan înregistrările părinte și copil.



FIGURA 7.4

Access generează o casetă de dialog cu un mesaj de eroare la încălcarea ștergerilor restricționate.

Introducerea restricționată a datelor

Introducerea restricționată a datelor înseamnă că o înregistrare copil nu poate referi un părinte inexistent. Înregistrările copil care referă un părinte inexistent sunt denumite și înregistrări orfane. Introducerea restricționată a datelor este obligatorie pentru integritatea referențială.

Figura 7.5 prezintă caseta de dialog cu mesajul de eroare, generat de Access la încălcarea introducerii restricționate a datelor.

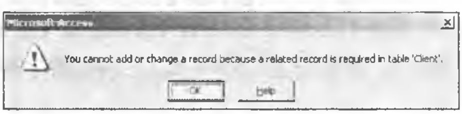


FIGURA 7.5

Access generează o casetă de dialog cu un mesaj de eroare la încălcarea introducerii restricționate a datelor.

Ștergerile în cascadă

Ștergerile în cascadă înseamnă că, dacă este ștearsă o înregistrare părinte, atunci toate înregistrările copil sunt șterse automat. Figura 7.6 prezintă opțiunile care trebuie configurate pentru a activa acest tip de restricții privind integritatea referențială. Când veți încerca să ștergeți o înregistrare părinte, Access va genera o casetă de dialog care vă informează că vor fi șterse și înregistrările înrudite dacă veți continua operațiunea de ștergere (vedeți figura 7.7).

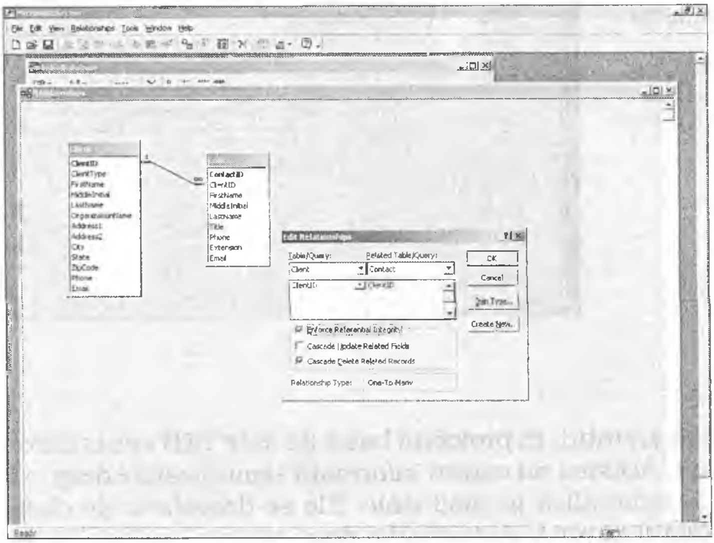


FIGURA 7.6 Bifarea opțiunii Cascade Delete Related Records duce la activarea ștergerilor în cascadă.

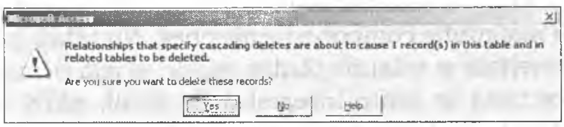


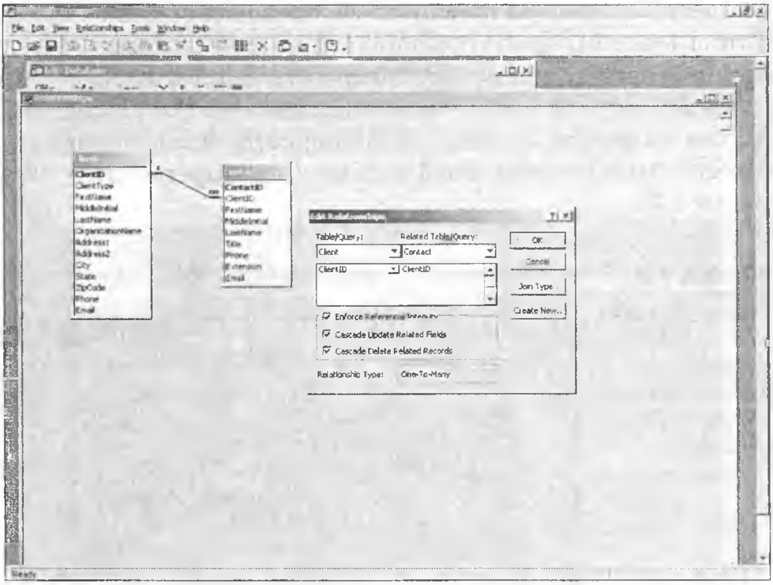
FIGURA 7.7 Înainte de ștergerea înregistrărilor înrudite, Access generează o casetă de dialog pentru ca utilizatorul să confirme continuarea operației.

Actualizările în cascadă

Actualizările în cascadă înseamnă că dacă se modifică cheia primară a unei înregistrări părinte, este actualizată și cheia externă a tuturor înregistrărilor copil. Pentru a activa această restricție privind integritatea referențială, trebuie să bifați caseta de validare Cascade Update Related Records. Figura 7.8 prezintă setarea necesară pentru activarea opțiunii cu acest nume.

FIGURA 7.8

Prin bifarea casetei de validare Cascade Update Related Records veți activa actualizarea în cascadă a înregistrărilor.



Dacă vă amintiți, în proiectul bazei de date TEB sunt utilizate chei primare surogat. Acestea nu conțin informații semnificative despre înregistrările pe care le indentifică în mod unic. Ele se deosebesc de cheile semnificative prin faptul că pot fi alcătuite din două sau mai multe câmpuri. De exemplu, o cheie primară poate fi alcătuită din combinația între numele de familie, prenume, oraș și stat. Actualizările în cascadă rezolvă situația în care se modifică una sau mai multe componente ale cheii. Nu uitați, pentru a suporta integritatea referențială și relațiile dintre tabele, cheia primară a părintelui trebuie să fie preluată în toate înregistrările copil, altfel sistemul nu va funcționa. Avantajul cheilor surogat este, evident, faptul că nu trebuie să vă faceți probleme dacă se modifică cheia primară. În consecință, nu trebuie să vă preocupe necesitatea actualizărilor în cascadă sau a complexității suplimentare. Dacă doriți să recapitulați aspectele discutate despre cele două tipuri de chei primare, opriți-vă puțin și recitiți capitolul 4.

Rezumat

În acest capitol am prezentat unul dintre conceptele cele mai importante privind bazele de date – integritatea referențială. Cele patru tipuri de restricții privind integritatea referențială discutate în acest capitol sunt: ștergerile restricționate, introducerea restricționată a datelor, ștergerile în cascadă și actualizările în cascadă. Integritatea referențială, care completează regulile de integritate la nivel de câmp și la nivel de înregistrare, reprezintă un instrument pe care ar trebui să-l utilizați pentru a asigura calitatea datelor

dintr-o bază de date. Ideea că o aplicație este la fel de bună ca și baza de date cu care lucrează a fost enunțată de mai multe ori în această carte. Continuând puțin această idee, calitatea unei baze de date este aceeași cu a datelor pe care le stochează și a informațiilor pe care le poate oferi. Partea a III-a a cărții se încheie cu acest capitol. Recapitulând, partea I a prezentat conceptele esențiale ale bazelor de date. Partea a II-a v-a ghidat în procesul proiectării și construirii unei baze de date. Partea a III-a a tratat întreținerea bazei de date prin intermediul regulilor de validare și al integrității referențiale. Partea a IV-a are ca scop extragerea informațiilor dintr-o bază de date folosind limbajul SQL (Structured Query Language).

PARTEA A IV-A

EXTRAGEREA INFORMAȚIILOR DIN BAZA DE DATE – INTRODUCERE ÎN SQL



8

SQL – ELEMENTE GENERALE

În acest capitol:

- Prezentare generală a limbajului SQL
- Declarația `Select`
- Rezultatele primei filtrări: utilizarea clauzei `Where`
- Prima organizare a rezultatelor: utilizarea clauzei `Order By`
- Combinarea mai multor tabele: utilizarea declarației `Join`
- Combinarea mai multor seturi de rezultate: utilizarea clauzei `Union`
- Rezumat

Scopul primei jumătăți a acestei cărți a fost proiectarea și construirea bazei de date. A doua jumătate, care începe cu acest capitol, se concentrează asupra funcționării acesteia. Nu uitați, valoarea bazei de date se măsoară prin utilitatea informațiilor care pot fi extrase din ea. Acest capitol și capitolul următor prezintă un nou limbaj, numit *SQL* (*Structured Query Language*). SQL este un limbaj independent de bazele de date care permite interogarea datelor și efectuarea operațiilor numite pe scurt *CRUD* (*Create, Update and Delete* – creare, actualizare și ștergere). Întrucât limbajul nu depinde de bazele de date, construcțiile sale, pe care le veți învăța, sunt aplicabile pentru SQL Server, Oracle, Informix, Sybase și DB2, pentru a menționa numai câteva aplicații pentru baze de date.

Prezentare generală a limbajului SQL

La fel cum pentru limbajele de programare precum Basic, C, C++, Java și Pascal există mai multe versiuni, așa și pentru SQL există mai multe versiuni. Ultima dintre ele este numită ANSI-SQL 1992. *ANSI* este prescurtarea de la *American National Standard Institute*. Ultima versiune standard a limbajului SQL a fost codificată în 1992. Un standard nu este decât o specificare pentru modul de funcționare al unui obiect. Dacă produsul pentru baze de date cu care lucrați suportă standardul ANSI, puteți fi sigur că vor fi suportate elementele esențiale ale limbajului. Aceasta nu înseamnă că o anumită bază de date trebuie să suporte toate părțile standardului, ci mai degrabă, de exemplu, că declarația Select din Oracle va funcționa la fel ca și în SQL Server.

Versiunile anumitor produse pentru baze de date se extind adesea dincolo de standard. De exemplu, limbajul SQL pentru server este denumit *T-SQL*, iar pentru Oracle este denumit *PL/SQL*. Fiecare dintre acestea este numit superset al standardului ANSI. Cu alte cuvinte, versiunile T-SQL și PL/SQL ale standardului ANSI nu suportă numai standardul ANSI-92, ci și construcțiile limbajului specifice bazei de date. Versiunea de SQL pentru Access respectă îndeaproape standardul ANSI. De fapt, SQL este format din trei sublimbaje:

- **DDL** – Data Definition Language (limbajul de definire a datelor);
- **DML** – Data Manipulation Language (limbajul de manipulare a datelor);
- **DCL** – Data Control Language (limbajul de control al datelor).

Acest capitol și cel care urmează se vor concentra asupra sublimbajului DML, cel pe care, probabil, îl veți folosi cel mai des. Sublimbajul DML este alcătuit din comenzile `Select`, `Insert`, `Update` și `Delete`. Acest capitol va studia numai comanda `Select` și clauzele asociate ei, iar capitolul următor va studia utilizarea avansată a comenzilor `Select`, `Insert`, `Update` și `Delete`. Cât despre DDL, în majoritatea cazurilor veți utiliza instrumentele grafice pentru crearea tabelelor și coloanelor. Access nu suportă sublimbajul

DCL. Cu toate acestea sublimbajele DDL și DCL vor fi prezentate pe scurt în secțiunea următoare.

Lucrul interactiv cu SQL în Access

Unele produse pentru baze de date au o fereastră care permite utilizatorului să scrie în mod interactiv comenzi în SQL. Access are și el o astfel de fereastră. Totuși, pentru a o folosi, trebuie să depășiți câteva obstacole.

În majoritatea cazurilor, veți utiliza fereastra SQL în Access pentru a crea interogări SQL Select. Așa cum veți vedea, există în SQL construcții care vă permit să definiți și obiecte ale bazei de date. Deși fereastra SQL din Access nu a fost proiectată în acest scop, ea suportă funcțiile Data Definition Language din SQL.

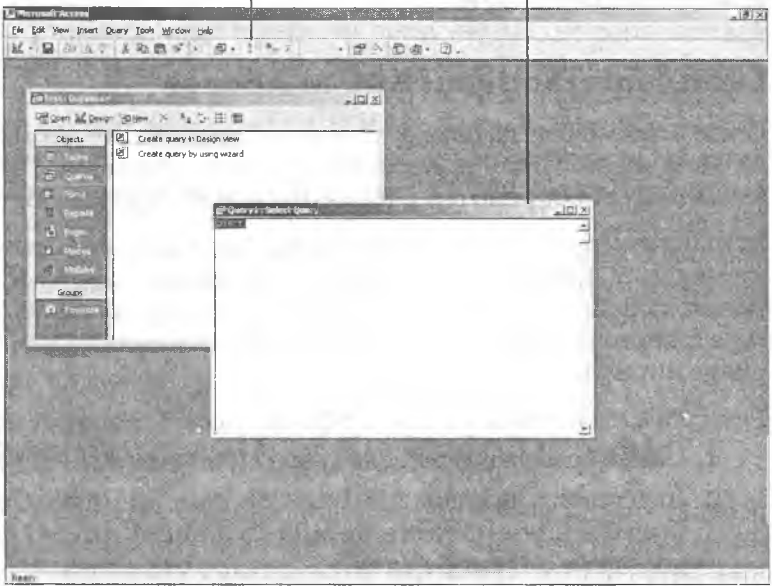
Pentru a avea acces la fereastra SQL din Access, parcurgeți pașii următori:

1. Lansați programul Access și alegeți crearea unei noi baze de date.
2. În fereastra principală a bazei de date, din lista Objects, selectați Queries. Lista Objects se află în panoul din stânga al ferestrei principale a bazei de date.
3. În panoul din dreapta, selectați opțiunea pentru crearea unei interogări în vizualizarea Design.
4. Închideți caseta de dialog Show Table, care vă permite să adăugați tabele în interogarea Select. Nu există tabele, pentru că baza de date este goală!
5. După ce închideți caseta de dialog Show Table, alegeți din bara de instrumente vizualizarea SQL. Aceasta este prima opțiune din stânga, etichetată SQL.
6. După ce parcurgeți acest pas, fereastra de proiectare a înregistrării va dispărea, apoi va apărea fereastra Select Query. Pentru a rula o anumită înregistrare, trebuie să executați clic pe butonul Run Query, de pe bare de instrumente Query. Acest buton este reprezentat sub forma unui semn de exclamație. Figura 8.1 prezintă rezultatele pe care ar trebui să le obțineți.

Pe parcursul acestui capitol, veți întâlni serii de pași care încep cu deschiderea unei ferestre SQL. Pentru aceasta veți urma pașii descriși mai sus. În majoritatea cazurilor, veși închide sesiunea de interogare fără a salva interogarea. Acest lucru este valabil mai cu seamă pentru comenzile din DDL. Vor fi salvate însă unele dintre comenzile limbajului DML. În orice caz, dacă doriți să salvați vreo comandă SQL, puteți face acest lucru executând clic pe butonul Save de pe bara principală de instrumente din Access.

FIGURA 8.1

Fereastra SQL vă permite să dați comenzi SQL în Access în mod interactiv.



Sublimbajele DDL și DCL din SQL

Access are un suport limitat față de sublimbajul DDL, și nu suportă sublimbajul DCL. Bazele de date cum sunt cele din SQL Server, Oracle și DB2, pentru a numi doar câteva, suportă total ambele sublimbaje.

DCL vă permite să oferiți anumitor utilizatori variate permisiuni, ca de exemplu Select, Insert, Update și Delete. Iată un exemplu de declarație SQL care oferă utilizatorului numit Bob permisiunea de selectare a tabelului Customers:

```
Grant select
      On Customers
      To Bob
```

În această situație, când Bob se conectează la baza de date, i se permite să execute următoarea declarație SQL:

```
Select *
      From Customers
```

Deși Access suportă și oferă utilizatorului interfețe specializate pentru a suporta o astfel de schemă de securitate, metoda de control și definire a permisiunilor de securitate nu a fost implementată prin sublimbajul DCL. Detaliile privind implementarea funcțiilor de securitate în Access depășesc scopul acestei cărți.

Pentru a obține mai multe informații despre implementarea securității în Access, consultați fișierul Help.

Sublimbajul DDL acționează asupra tabelelor și coloanelor. Puteți adăuga modifica sau renunța la definiția unui tabel. Ca și în cazul tabelelor, puteți adăuga, modifica sau renunța și la definiția unei coloane. Ca și în cazul sublimbajului DCL, bazele de date din SQL Server, Oracle și DB2 oferă suport deplin pentru DDL. Spre deosebire de sublimbajul DCL, Access suportă limitat sublimbajul DDL. Rulând codul următor în fereastra SQL, veți crea un tabel numit Customers:

```
Create Table Customers
```

Desigur, când creați tabele, creați în același timp și coloane. Codul următor creează un tabel numit Customers, care conține două câmpuri, `custid` și `name`:

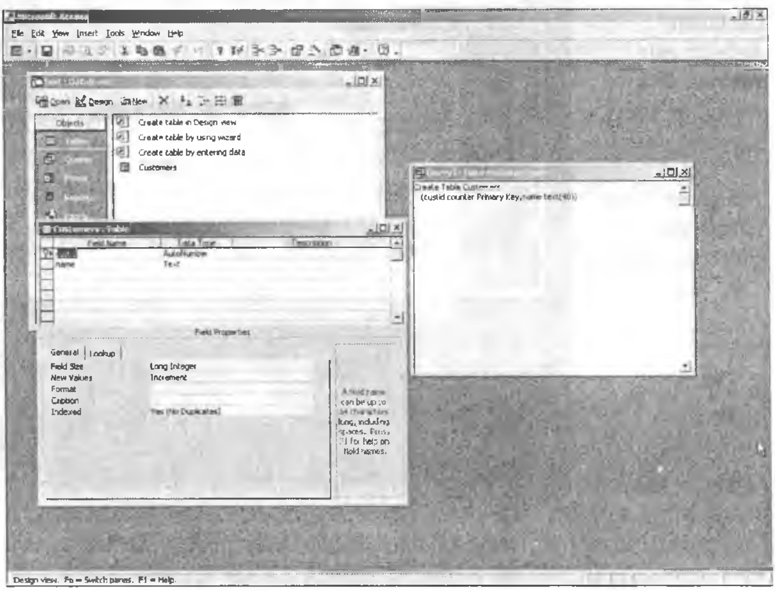
```
Create Table Customers
      (custid counter Primary Key,name text(40))
```

Așa cum arată și codul, câmpul `custid` este de tip AutoNumber și are rol de cheie primară. Câmpul `name` are tipul text, cu lungimea de 40 de caractere. Figura 8.2 prezintă rezultatele rulării declarației Create Table. În afara efectuării acestor operații, puteți modifica și structura tabelului. Codul următor modifică lungimea câmpului `name` din 40 în 50 de caractere:

```
Alter Table Customers
      Alter name text(50)
```

FIGURA 8.2

Access oferă suport limitat pentru sublimbajul DDL din SQL.



Access oferă suport limitat sublimbajului DDL și în privința definirii relațiilor dintre tabele și a regulilor de validare la nivel de câmp și la nivel de înregistrare. Datorită acestui suport limitat și modului în care a fost proiectat programul Access, este foarte probabil că veți utiliza instrumentele grafice pentru întreținerea tabelelor și coloanelor. În acest stadiu, tot ce contează este să știți ce este DDL și că posibilitățile sale sunt limitate în Access, dar nu și în alte programe pentru baze de date.

După această scurtă prezentare a limbajului SQL și a sublimbajelor sale, vom trece la sublimbajul DML (Data Manipulation Language), pe care îl veți utiliza în majoritatea situațiilor când lucrați în SQL. DML este alcătuit din comenzile `Select`, `Insert`, `Update` și `Delete`. Unele dintre aceste comenzi au clauze adiționale care le influențează funcționarea. Să punem la lucru baza de date TEB!

Declarația `Select`

Declarația `Select` este, dintre toate declarațiile SQL, cea pe care o veți utiliza în majoritatea cazurilor. Interogările au sopul unic de a extrage informații din baza de date. Pentru a îndeplini acest scop, veți utiliza declarația `Select`. Figura 8.3 prezintă o declarație `Select` foarte simplă.

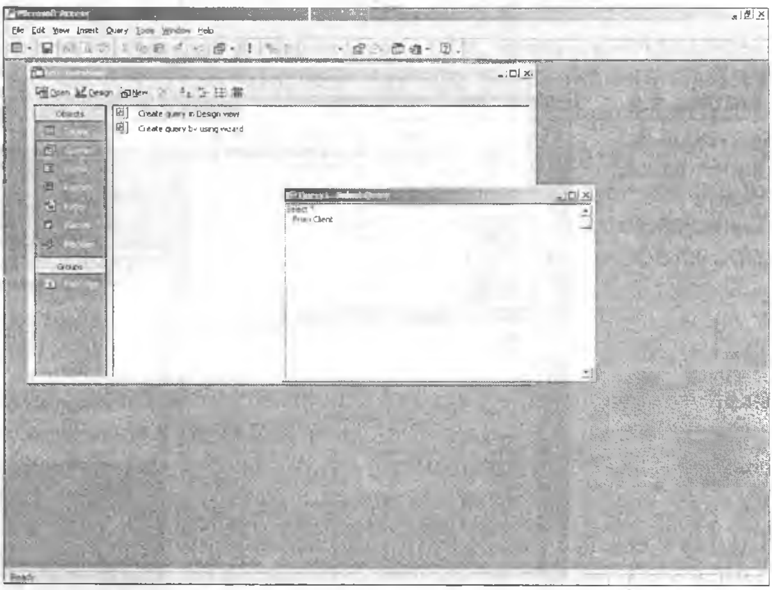
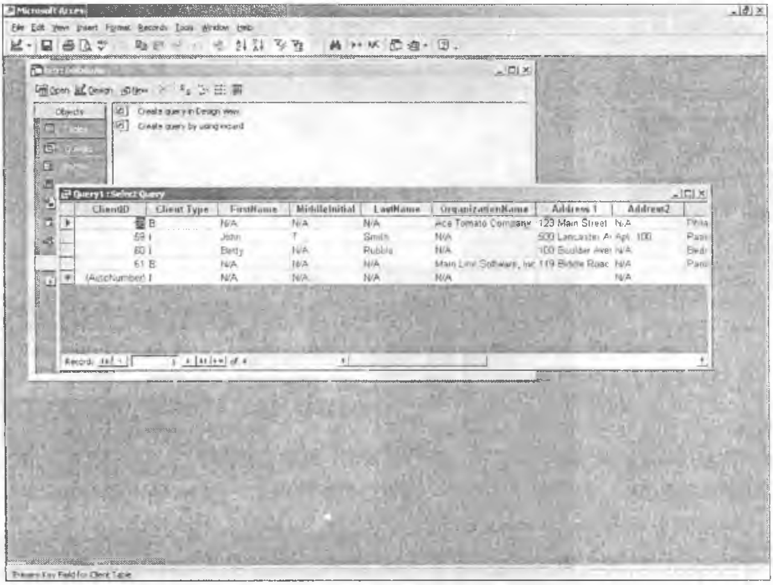


FIGURA 8.3
Simbolul * dintr-o declarație `Select` este metoda de a indica pe scurt bazei de date că trebuie să returneze toate câmpurile dintr-un tabel.

Când rulați o declarație în fereastra SQL, fereastra este înlocuită cu vizualizarea Datasheet. Dacă vă amintiți discuțiile anterioare, când deschideți un tabel pentru a privi datele din el, acesta este prezentat în vizualizarea Datasheet. Când deschideți tabelul Client pentru a privi datele, este ca și cum ați emite declarația SQL `Select * From Client`. Figura 8.4 prezintă rezultatele care vor fi afișate.

FIGURA 8.4

Când deschideți un tabel, veți obține vizualizarea Datasheet cu rezultatele unei interogări.



Atenție

Fiți atent când modificați datele într-o înregistrare din vizualizarea Datasheet. În funcție de circumstanțe, acest lucru va duce la modificarea datelor stocate în tabele. Dacă utilizați instrumentele de proiectare grafică a interogărilor, rezultatele pot fi marcate cu statutul read-only (numai citire). Proiectarea interogărilor este discutată detaliat în capitolul 10.

Selectarea unor anumite câmpuri

În majoritatea cazurilor, nu veți dori să selectați toate câmpurile dintr-un tabel, ci numai câteva dintre ele. De exemplu, ClientID nu are semnificație în afara bazei de date. Într-un raport simplu care prezintă o listă cu clienți, estre posibil ca elemente precum numărul de telefon sau adresa de e-mail să nu fie necesare. În loc să scrieți simbolul*, puteți scrie câmpurile respective, în ordinea în care doriți ca ele să apară în setul de rezultate.

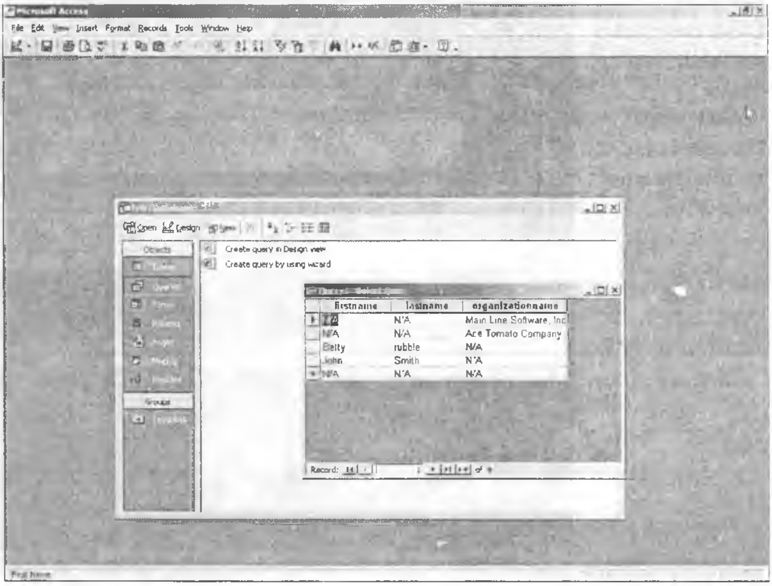
Pentru a exemplifica, să presupunem că doriți să adunați informații despre numele de familie și prenumele clientului sau despre numele firmei, după caz. Următoarea declarație `Select` în SQL va rezolva problema:

```
SELECT firstname, lastname, organizationname
      FROM Client;
```

Figura 8.5 prezintă rezultatele acestei interogări.

FIGURA 8.5

Interogările `Select` în SQL vă permit să colectați informații din anumite coloane.



Redenumirea câmpurilor folosind clauza `As`

Puteți alege câmpurile incluse în interogare și ordinea acestora, dar puteți și să atribuiți alte nume câmpurilor selectate. Veți realiza acest lucru folosind clauza As. Priviți cu atenție următoarea declarație `Select` în SQL:

```
SELECT firstname As fname,
            lastname As lname,
            organizationname As org
      From Client
```

Figura 8.6 prezintă rezultatele acestei interogări.

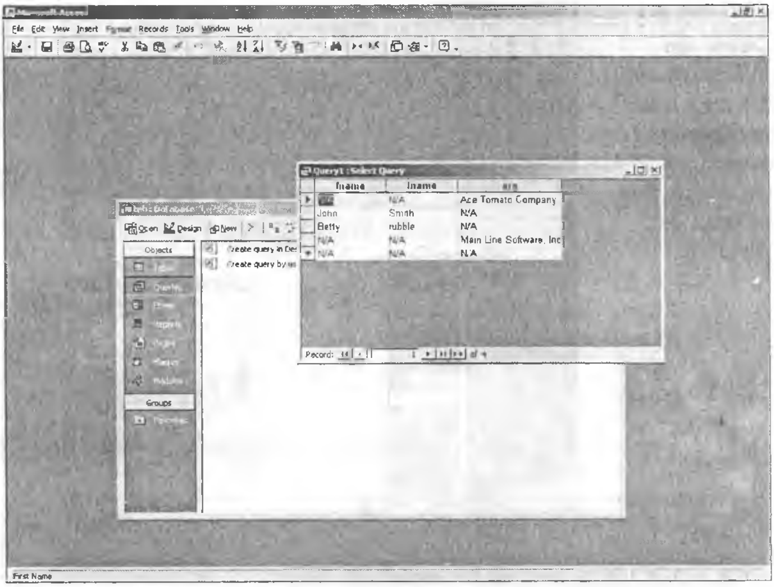


FIGURA 8.6
Clauza As vă permite să redenumiți un câmp în rezultatele interogării.

Combinarea câmpurilor de date

Uneori veți dori să combinați date din mai multe câmpuri și să returnați datele respective într-un singur câmp din setul de rezultate. De exemplu, în loc de a returna numele de familie și prenumele în câmpuri separate, doriți ca aceste date să fie combinate într-un singur câmp denumit *name* (nume). Declarația `Select` în SQL care urmează realizează acest obiectiv:

```
SELECT firstname + " " + lastname As name,
            organizationname As org
      From Client
```

Figura 8.7 prezintă rezultatele acestei interogări.

Există și un risc în combinarea datelor, numită adesea *concatenare*. Datele pe care le concatenați trebuie să aibă același tip. De obicei, veți combina date de tip text. Datele de tip numeric și cele de tip date/time (data calendaristică/ora) sunt adunate aritmetic dacă este utilizat operatorul +. Dacă doriți să concatenați datele de aceste tipuri, nu să le adunați aritmetic, trebuie să le convertiți la tipul de date text, apoi să combinați datele convertite. Pentru a ilustra diferența, priviți următorul cod SQL:

```
SELECT 1+2+3 as combinedfield
      FROM Client;
```

FIGURA 8.7

Datele din diferite câmpuri ale unui tabel pot fi combinate pentru a forma un singur câmp în setul de rezultate.

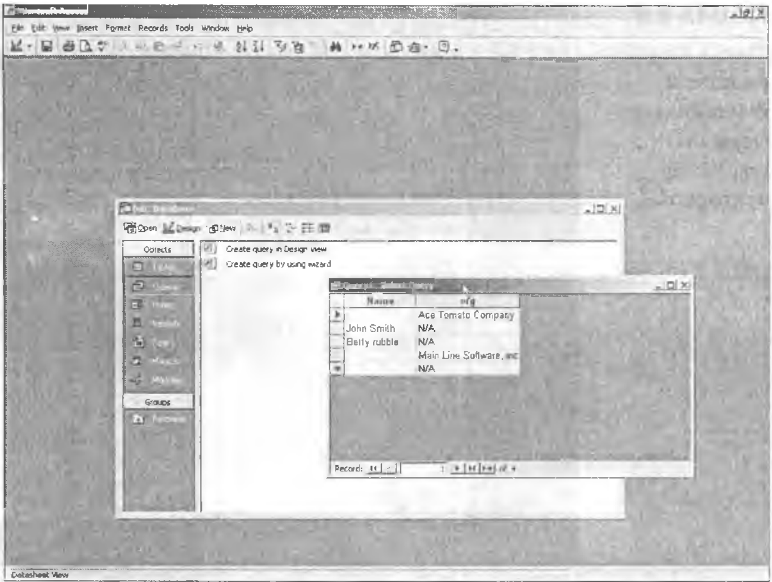
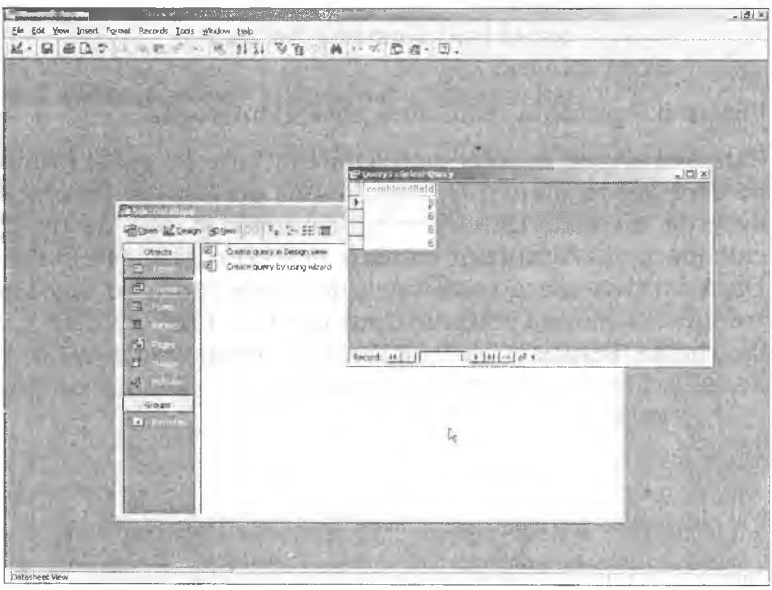


Figura 8.8 prezintă rezultatele acestei interogări. După cum vedeți, puteți referi nu numai câmpurile actuale într-o declarație Select, dar și date statice de tip numeric, dată calendaristică sau text (aceste valori statice sunt numite *literale*). În funcție de motorul pentru baze de date pe care îl utilizați, puteți referi orice funcție pe care motorul pentru baze de date o înțelege.

FIGURA 8.8

Într-o declarație `Select` pot fi utilizate datele statice de tip numeric, dată calendaristică și text.



În acest caz, numerele 1, 2 și 3 sunt adunate aritmetic. Motivul pentru care vedeți 4 înregistrări, fiecare cu valoarea 6, este că pentru fiecare înregistrare este returantă altă înregistrare în setul de rezultate.

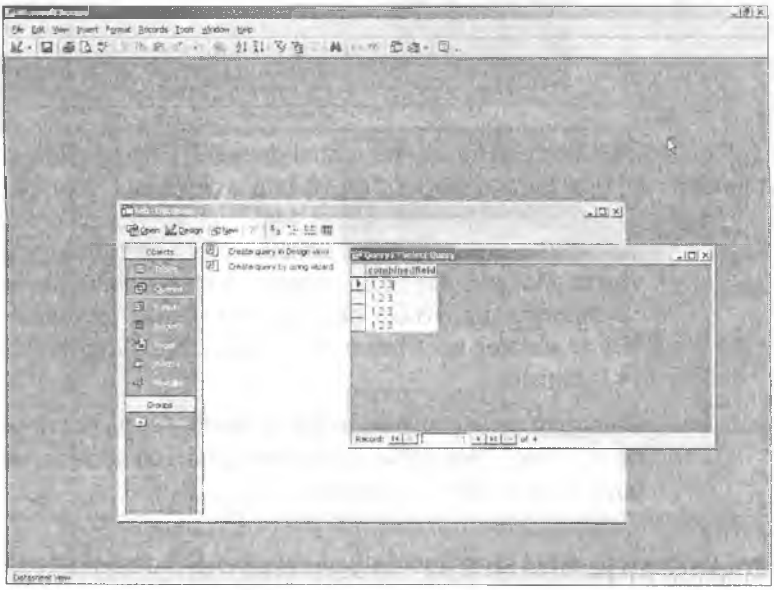
Pe de altă parte, dacă doriți ca valoarea returnată să fie 123, trebuie să convertiți toate elementele în text, apoi să combinați valorile de tip text. Desigur, ați putea să selectați foarte ușor șirul “123”; dar astfel nu aveți șansa de a vedea cum pot fi convertite în text valorile numerice! Codul următor vă duce la rezultatele dorite:

```
SELECT Str(1)+Str(2)+Str(3) as combinedfield
      FROM Client;
```

Figura 8.9 prezintă rezultatele interogării.

FIGURA 8.9

Pentru combinarea datelor pot fi utlizate funcțiile dintr-o bază de date, cum este funcția `Str()`.



Declarațiile de tip Top *n* Select

Ultimul subiect pe care îl vom discuta despre declarația `Select` este cel al declarațiilor de tip `Top` *n* `Select`. Acestea mai sunt numite și interogări de tip `Top` *n*. De multe ori veți dori să aflați numai prima înregistrare, primele 10 sau primele 100 de înregistrări dintr-un set de rezultate. Utilizând exemplul precedent, în care numărul de rânduri din setul de rezultate depinde de numărul de rânduri din tabelul de bază, o interogare de tip `Top` *n* poate controla numărul de rânduri din setul de rezulate. Codul următor forțează setul de rezultate să aibă un singur rând:

```
SELECT Top 1 Str(1)+Str(2)+Str(3) as combinedfield
      FROM Client;
```

Figura 8.10 prezintă rezultatele acestei interogări.

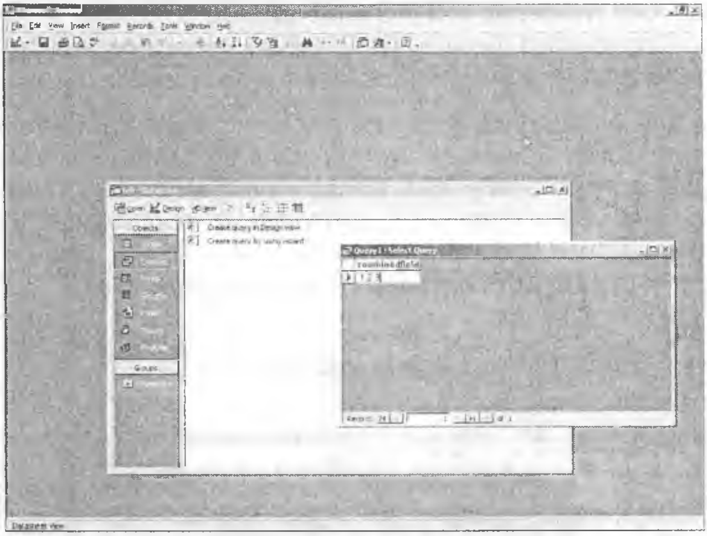


FIGURA 8.10
O interogare de tip `Top` *n* controlează numărul de rânduri care apar în setul de rezultate.

Atenție

Deși doriți să fie afișate numai primele 10 înregistrări, este foarte posibil să fie returnate mai mult de 10. Iată un exemplu: doriți să fie afișate primele 10 produse, în funcție de preț. Mai mult, al zecelea produs are prețul 10$, dar există 5 produse care au același preț. Câte înregistrări vor fi returnate? Veți obține cel puțin 10 înregistrări. A zecea dintre ele este primul produs cu prețul de 10$. Totuși, deoarece mai există 4 produse care au prețul de 10$, și acestea vor fi returnate. În această situație ipotetică sunt returnate 14 înregistrări.

Care este concluzia? Interogările de tip `Top` *n* pot returna mai mult de *n* rânduri dacă există mai multe înregistrări cu aceeași valoare a expresiei după care se face ordonarea (`order by`).

După ce ați învățat cum să extrageți și să manipulați datele, pasul următor este filtrarea și organizarea lor. Interogările de tip `Top` *n* sunt mecanisme de filtrare. Totuși, restricția `Top` *n* este aplicată *după* aplicarea filtrului specificat în clauza `Where`. Filtrarea și organizarea datelor vor fi tratate în următoarele două secțiuni.

Rezultatele primei filtrări: utilizarea clauzei `Where`

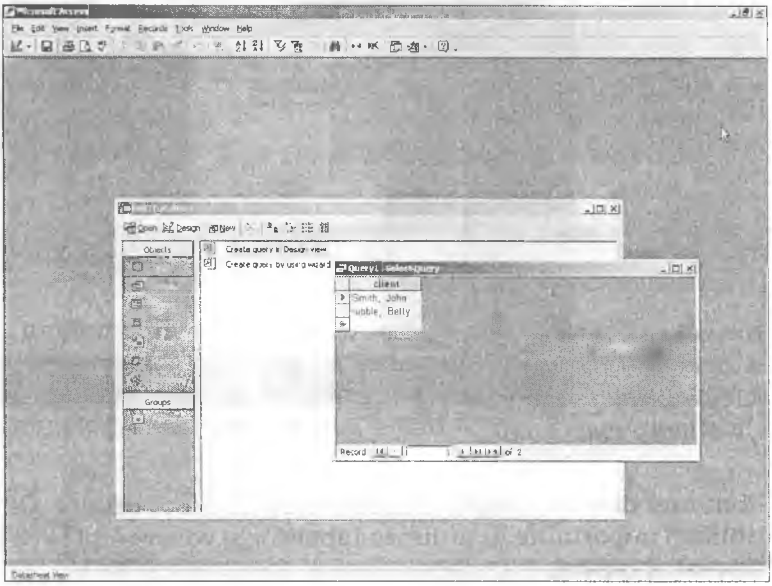
În practică, veți utiliza întotdeauna un filtru în interogările dumneavoastră. Vom extinde unele dintre exemplele anterioare clasificându-le după un anumit tip de client. Dacă vă amintiți, clienții pot fi persoane fizice sau firme. Câmpul ClientType arată ce tip corespunde fiecărui client. Pentru a ilustra funcționarea clauzei `Where`, să extragem datele cu prenumele și numele de familie pentru clienții persoane fizice.

```
SELECT lastname + ", " + firstname As Client
      From Client
      Where ClientType = "I"
```

Au mai fost făcute câteva modificări. Ați observat că ordinea între prenume și numele de familie a fost inversată și că au fost separate de virgulă? Funcționarea clauzei Where este destul de simplă. Puteți utiliza unul sau mai multe criterii. Figura 8.11 prezintă rezultatele interogării.

FIGURA 8.11

Clauza Where vă permite să excludeți datele nerelevante.



Să extindem acest exemplu, presupunând că doriți să extrageți toți clienții persoane fizice al căror nume de familie începe cu S. Codul următor rezolvă acest lucru:

```
SELECT lastname + ", " + firstname As Client
      From Client
      Where ClientType = "I" And lastname Like "S*"
```

Acest exemplu necesită studierea atentă a câtorva aspecte. Mai întâi, așa cum am precizat anterior, puteți combina mai multe criterii în clauza Where. Apoi, observați utilizarea cuvântului Like în acest caz. În acest exemplu, doriți să extrageți toți clienții persoane fizice al căror nume de familie începe cu litera S. Veți spune că, în acest caz, criteriile sunt slabe. De obicei, împreună cu operatorul de comparație Like este utilizat un caracter wildcard. În Access, acesta are forma unui unui asterisc (*). În SQL Server, caracterul wildcard este semnul procentelor (%). Dacă ați fi folosit semnul egal, Access

ar fi căutat înregistrările pentru care numele de familie este `S*`. Figura 8.12 prezintă rezultatele interogării cu criterii multiple, în care apare și semnul wildcard.

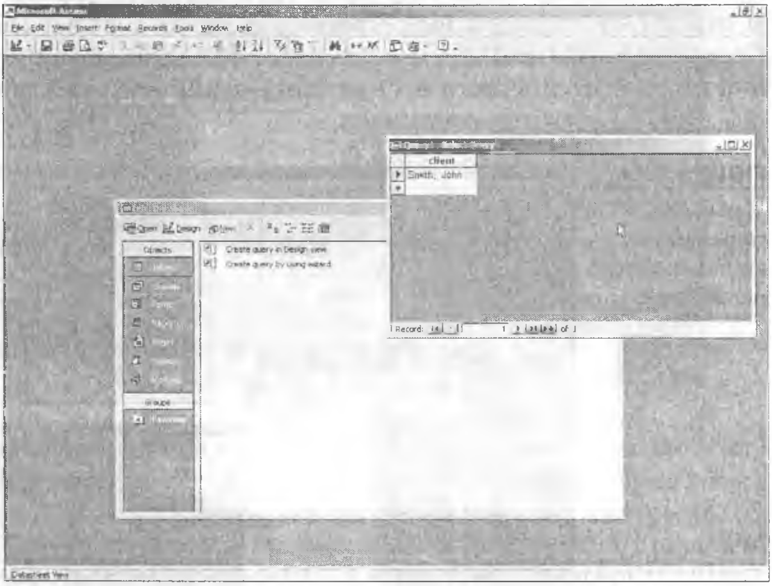


FIGURA 8.12

Utilizarea criteriilor wildcard permite căutarea după criterii slabe.

Vom mai discuta despre clauza `Where` în acest capitol. Clauza `Where` are utilizări importante în uniunea tabelelor și vom reveni la ea în altă secțiune a acestui capitol.

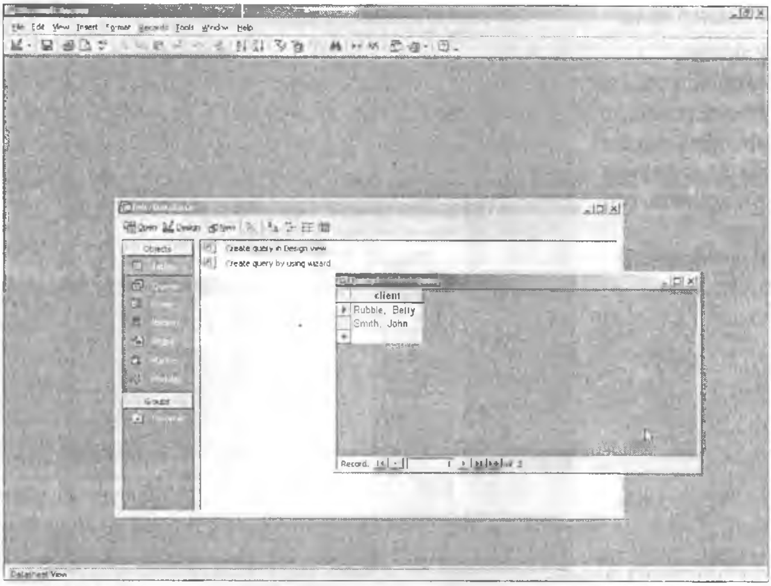
Prima organizare a rezultatelor: utilizarea clauzei `Order By`

La fel cum utilizați întotdeauna clauza `Where` pentru a filtra date nerelevante, veți utiliza clauza `Order By` pentru sortarea datelor din setul de rezultate. Mai mult, pe baza câmpurilor pe care le alegeți pentru a sorta datele, veți dori să le sortați ascendent, descendent sau după un criteriu combinat. Toate aceste operații vor fi efectuate prin intermediul clauzei `Order By`. Pentru a ilustra funcționarea ei, priviți codul următor:

```
SELECT lastname + ", " + firstname As Client
      From Client
      Where ClientType = "I"
      Order By lastname
```

Figura 8.13 prezintă rezultatele acestei interogări.

FIGURA 8.13
Clauza Order By vă permite să ordonați datele din setul de rezultate.



În mod implicit, indiferent de câmpul pe care îl alegeți, sortarea se face în ordine crescătoare (ascendent). Dacă doriți să sortați rezultatele în ordine descrescătoare (descendent) trebuie să solicitați explicit acest lucru. În afară de calificativul pentru ordine descendentă, codul următor este identic cu cel precedent:

```
SELECT lastname + ", " + firstname As Client
      From Client
      Where ClientType = "I"
      Order By lastname Desc
```

Figura 8.14 prezintă rezultatele acestei interogări.

Sortarea după mai multe criterii

Să presupunem că doriți să realizați o listă cu clienții persoane fizice, în care sunt prezentate numele, orașul și statul fiecărui client. În plus, doriți să sortați datele după stat, oraș și numele de familie. De obicei, atunci când sortați după mai multe câmpuri, ordinea în care apar acestea în lista de selecție va fi aceeași cu ordinea în care vor fi sortate; acest lucru nu este însă obligatoriu. Dar rezultatele sunt mai ușor de citit. Codul următor are ca rezultat afișarea statului, orașului și a unei combinații între prenume și numele de familie. În plus, rezultatele sunt sortate după stat, oraș și numele de familie:

```
SELECT state,city,lastname + ", " + firstname As Client
      From Client
      Where ClientType = "I"
      Order By state,city,lastname
```

Figura 8.15 prezintă rezultatele acestei interogări.

FIGURA 8.14

Folosind calificativul Desc, puteți specifica dacă doriți o sortare ascendentă sau una descendentă.

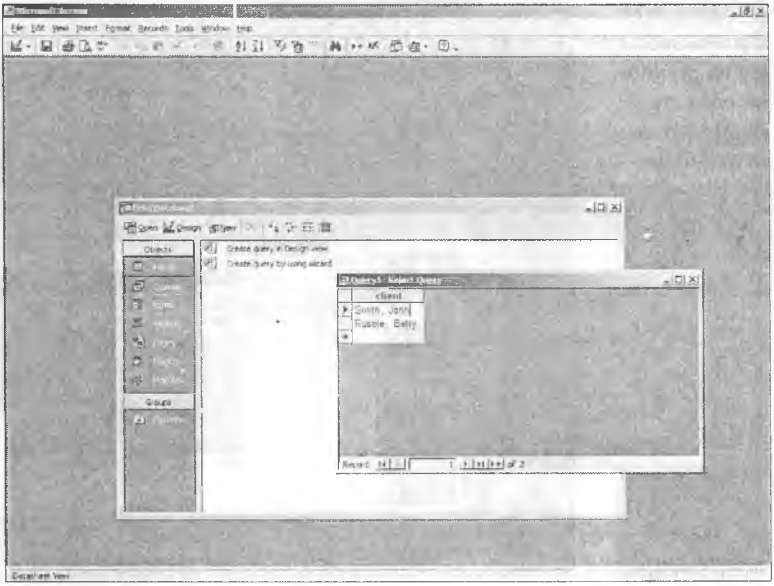
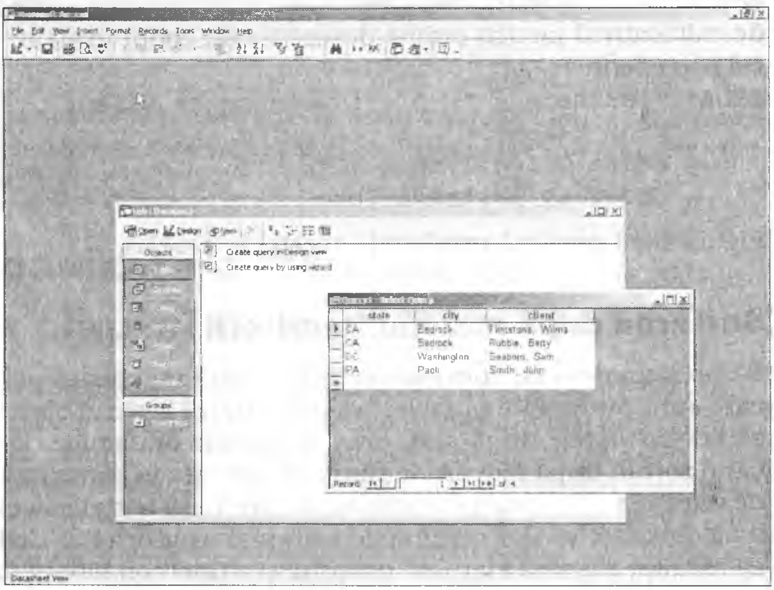


FIGURA 8.15

Puteți sorta setul de rezultate după mai multe câmpuri.



După cum vedeți, nu trebuie neapărat să alegeți un anumit câmp pentru ca el să fie utilizat într-o sortare. În această situație câmpul `lastname` nu apare ca atare în setul de rezultate. În schimb, el a fost combinat cu câmpul `firstname`. Cu toate acestea, puteți sorta după acest câmp, câtă vreme el este disponibil în tabelul de bază. În majoritatea situațiilor, totuși, datele utilizate pentru sortare vor fi disponibile în setul de rezultate, fie în câmpul corespunzător, fie combinate cu datele din alt câmp.

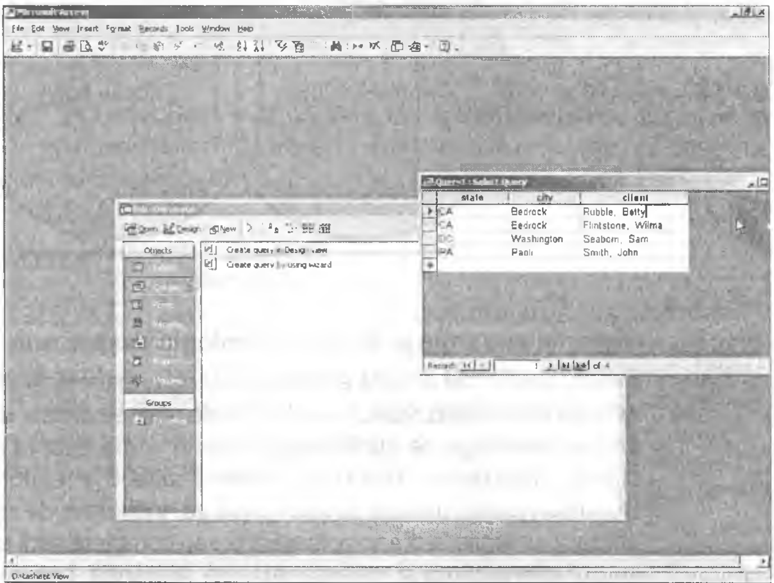
Să extindem exemplul precedent, alegând să sortăm după stat, după oraș în ordine descrescătoare, apoi după numele de familie în ordine descrescătoare. Codul următor realizează acest scop:

```
SELECT state,city,lastname + ", " + firstname As Client
      From Client
      Where ClientType = "I"
      Order By state,city Desc,lastname Desc
```

Figura 8.16 prezintă rezultatele acestei interogări.

FIGURA 8.16

În cadrul fiecărui câmp după care efectuați sortarea, puteți alege și sortarea descendentă.



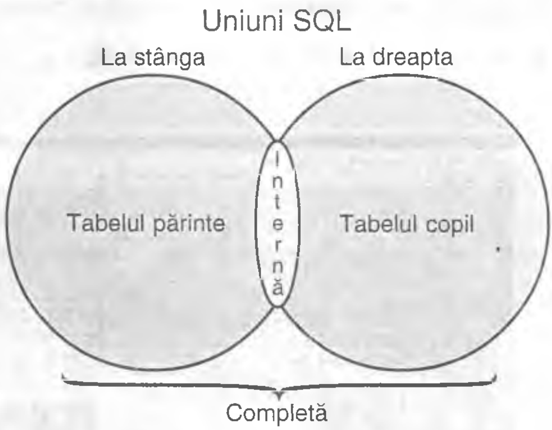
După cum vedeți, primul criteriu de sortare este statul. Totuși, dacă veți compara figurile 8.15 și 8.16 veți vedea că Betty Rubble și Wilma Flinstone au schimbat locurile în setul de rezultate. Nu uitați că sortările după oraș și numele de familie se fac descendent. Dar dacă Wilma Flinstone locuia în Copper Falls? Întrucât sortarea se face mai întâi după stat, apoi după oraș și în final după numele de familie, în acest caz Wilma Flinstone ar apărea din nou în setul de rezultate înainte de Betty Rubble. Este un bun exercițiu să vă jucați puțin cu un set mic de date, să le modificați, apoi să rulați din nou diferitele interogări pentru a vedea care sunt efectele.

Combinarea mai multor tabele: utilizarea declarației `Join`

În majoritatea situațiilor veți dori să extrageți date din mai multe tabele. De exemplu, este posibil să vi se solicite crearea unei liste cu clienții persoane juridice (firme) și contactele corespunzătoare acestora. Acesta este cadrul de bază care va fi folosit pentru a introduce conceptul de uniune a tabelelor. Există patru tipuri de uniune: internă (*inner*), externă la stânga (*left outer*), externă la dreapta (*right outer*) și complet externă (*full outer*). Figura 8.17 oferă o ilustrare grafică a funcționării, în relație cu tabelele unite, a diferitelor tipuri de uniune. Pe parcursul acestei discuții veți reveni mereu la figura 8.17. Pentru a exemplifica funcționarea tipurilor de uniune, vom folosi tabelele Client și Contact.

FIGURA 8.17

Există patru tipuri de uniune între tabele: internă, la stânga, la dreapta și completă.



Notă

Scurtă observație privind terminologia despre uniunea tabelelor

Pentru tipurile de uniune externă la stânga, externă la dreapta și complet externă utilizarea cuvântului *externă* este adesea omisă; dacă cineva spune uniune la stânga, se subînțelege că este vorba despre uniune externă la stânga. Utilizarea cuvântului *externă* este o chestiune de preferințe. Dumneavoastră trebuie să știți numai că, indiferent de sursa de informare utilizată, uniunile la dreapta, la stânga sau completă sunt identice, respectiv, cu uniunile externă la dreapta, externă la stânga, complet externă.

Înainte de a trece mai departe, să precizăm care este interogarea de bază pe care o vom utiliza. În acest context vor fi luați în considerare numai clienții persoane juridice. Pentru a vedea cu ce începeți, interogarea următoare poate fi utilizată pentru a selecta toți clienții persoane juridice:

```
SELECT organizationname
      From Client
      Where ClientType = "B"
      Order By organizationname
```

Figura 8.18 prezintă lista clienților persoane juridice (firme), de la care veți porni.

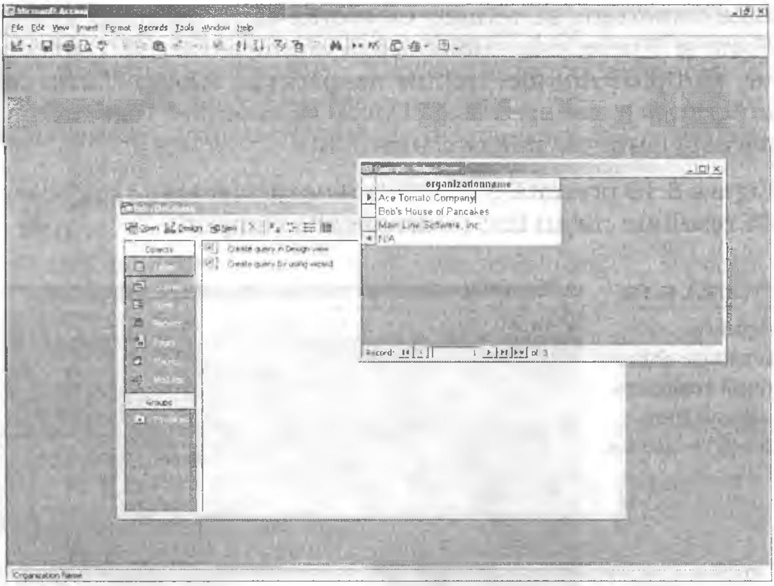


FIGURA 8.18

Lista completă a clienților care sunt firme din tabelul Client.

Acum, după ce ați realizat lista completă a clienților persoane juridice și aveți la dispoziție o interogare de bază cu care puteți lucra, putem discuta despre uniunea tabelelor în SQL.

Uniunile interne

Revenind la figura 8.17, puteți vedea că o uniune internă reprezintă intersecția între un tabel părinte și un tabel copil. Care este intersecția în acest caz? Dacă vă amintiți relația dintre tabelele Client și Contact, câmpul cheie primară `clientid` din tabelul Client este preluat ca și cheie externă în tabelul Contact. Din punct de vedere sintactic, clauza `Join` apare după clauza `From`. Codul următor ilustrează utilizarea clauzei `Join`:

```
SELECT organizationname
      From Client
        Inner Join Contact On (contact.clientid=client.clientid)
      Where ClientType = "B"
      Order By organizationname
```

Există și altă metodă pentru realizarea uniunii între tabele. În loc de a folosi sintaxa din blocul anterior de instrucțiuni, puteți utiliza sintaxa următoare:

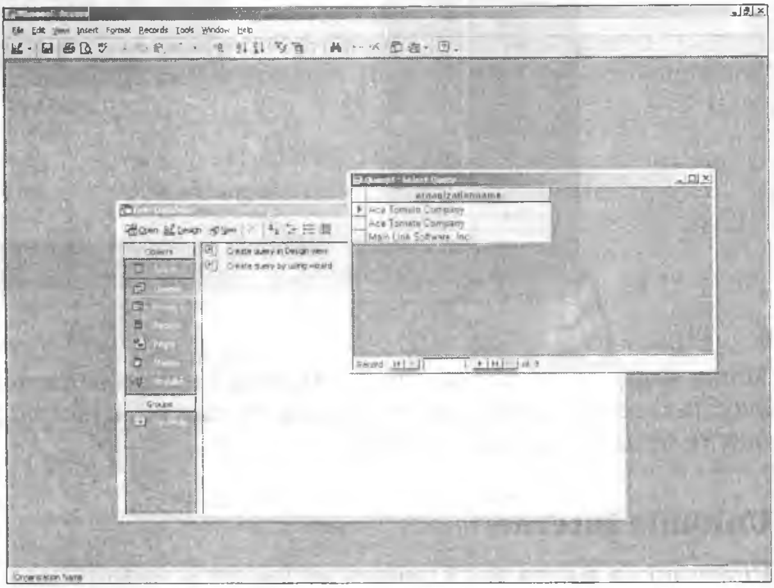
```
SELECT organizationname
      From Client,Contact
      Where ClientType = "B" And
            contact.clientid=client.clientid
      Order By organizationname
```

Este important să rețineți că atunci când includeți uniunea în clauzele From și Where, este subînțeleasă uniunea internă. Dacă doriți să realizați un alt tip de uniune, trebuie neapărat să folosiți clauza Join. Alternativa anterioară a fost prezentată numai ca exemplu. Toate exemplele de cod din această carte vor utiliza clauza Join.

Figura 8.19 prezintă efectul introducerii clauzei `Join`. Observați că în setul de rezultate nu au fost returnate alte câmpuri.

FIGURA 8.19

Setul de rezultate obținut după realizarea uniunii între tabelele Client și Contact.



Ar trebui să observați ușor două aspecte. Mai întâi, Ace Tomato Company apare de două ori. Apoi, firma Bob's House of Pancakes a dispărut din listă. Motivul dublei apariții a firmei Ace Tomato este acela că pentru ea există două contacte, iar motivul dispariției firmei Bob's House of Pancakes este faptul că pentru ea nu este definit nici un contact. Nu uitați, o uniune internă este rezultatul intersecției a două tabele. Cu alte cuvinte, o uniune internă va returna date dacă și numai dacă datele din tabelul părinte (din stânga) au înregistrări corespondente în tabelul copil (din dreapta). În acest exemplu, dacă identificatorul clientului din tabelul Client se regăsește în înregistrările din tabelul Contact, înregistrările corespondente sunt „condamnate să existe". În cazul firmei Ace Tomato Company, în listă apar două înregistrări corespondente, deoarece există două înregistrări în tabelul Contact.

Utilizarea clauzei `Distinct` pentru eliminarea datelor redundante

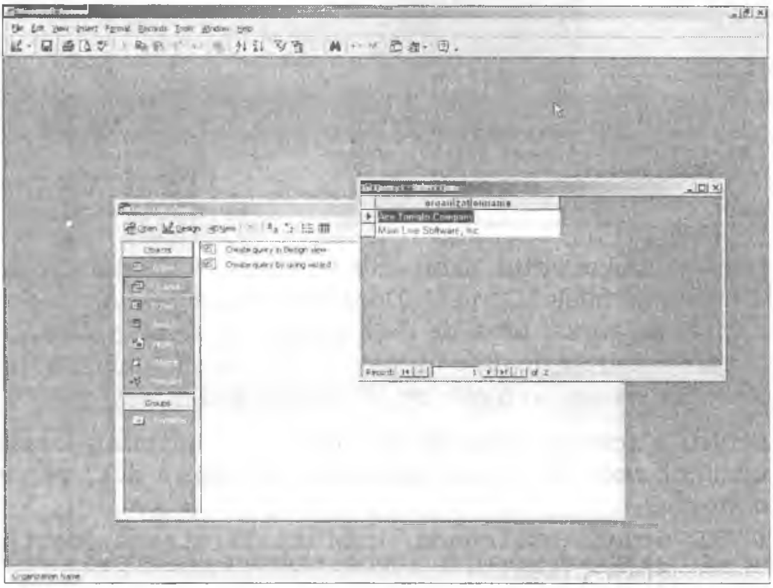
Ce ați zice dacă vi s-ar solicita o listă conținând numai acele firme care au date de contact? Categoric, trebuie să realizați o uniune internă, pentru a

elimina din listă clienții despre care nu aveți date de contact. În același timp, doriți ca fiecare client să apară în listă o singură dată. În secțiunea anterioară au fost prezentate interogările Top *n*. Totuși, în acest caz, o astfel de interogare nu va fi utilă. Ceea ce doriți este o altă listă (vedeți figura 8.20) și, din fericire, următorul bloc de instrucțiuni SQL vă oferă soluția perfectă:

```
SELECT Distinct organizationname
      From Client
        Inner Join Contact On (contact.clientid=client.clientid)
      Where ClientType = "B"
      Order By organizationname
```

FIGURA 8.20

Clauza `Distinct` face ca fiecare rând din setul de rezultate să fie unic.



Returnarea datelor din mai multe tabele

Continuând exemplul precedent, să extragem acum prenumele și numele de familie din tabelul Contact. Următorul bloc de instrucțiuni pare să realizeze acest obiectiv:

```
SELECT organizationname,firstname,lastname
      From Client
        Inner Join Contact On (contact.clientid = client.clientid)
      Where ClientType = "B"
      Order By organizationname
```

Figura 8.21 arată ce se întâmplă dacă încercați să rulați acest cod.

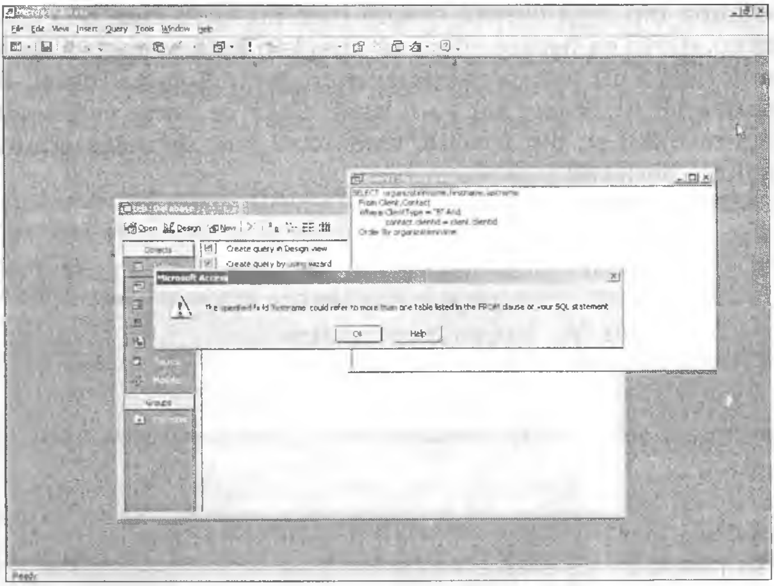


FIGURA 8.21
Dacă numele coloanelor din clauzele `From` și `Join` apar în cel puțin două tabele, trebuie să precizați tabelele din care sunt preluate pentru a rezolva ambiguitatea.

Dacă vă mai amintiți, câmpurile firstname și lastname există și în tabelul Client și în tabelul Contact. Dacă nu indicați bazei de date ce tabel trebuie utilizat ca sursă, baza de date trebuie să rezolve singură ambiguitatea. Deoarece bazele de date nu pot rezolva aceste ambiguități, veți obține un mesaj de eroare cum este cel din figura 8.21.

Pentru a rezolva ambiguitățile, trebuie să precizați tabelul dorit. Codul următor rezolvă această problemă, iar figura 8.22 prezintă rezultatele obținute.

```
SELECTorganizationname, contact.firstname, contact.lastname
    From Client
      Inner Join Contact On (contact.clientid = client.clientid)
    Where ClientType ="B"
    Order By organizationname
```

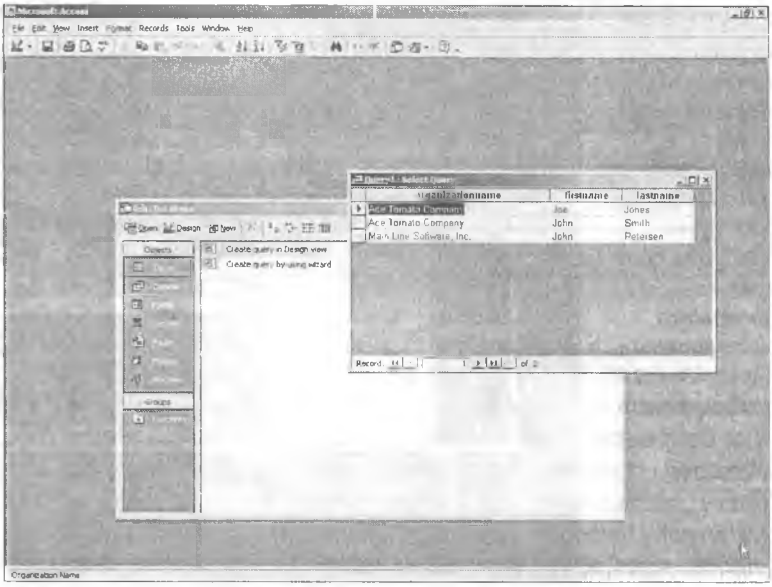
Unluni externe la stânga

Într-o uniune internă, înregistrările corespondente se află de o parte și de alta a uniunii. Dar dacă doriți să vedeți înregistrările din tabelul din stânga (tabelul părinte), indiferent dacă pentru ele există înregistrări corespondente în tabelul din dreapta (tabelul copil)? Obținerea acestor date este ușoară dacă utilizați o uniune externă la stânga. Adesea, *uniunea externă la stânga* este denumită, scurt, *uniune la stânga.* Următorul cod prezintă utilizarea uniunii la stânga:

```
SELECT organizationname, contact.firstname, contact.lastname
      From Client
        Left Outer Join On (contact.clientid = client.clientid)
Where ClientType="B"
Order By organizationname
```

Figura 8.23 prezintă rezultatele acestei interogări.

FIGURA 8.22
Când uniți tabelele, puteți extrage date din mai multe tabele în același set de rezultate.



În cazul firmei Bob's House of Pancakes, orice câmp care ar corespunde în tabelul Contact are valoare nulă. De ce? Deoarece nu există o înregistrare cu date de contact pentru Bob's House of Pancakes. Puteți considera că valorile nule sunt înlocuitori (placeholder) pentru datele care ar fi fost acolo dacă ar fi existat o înregistrare cu date de contact despre clientul respectiv.

Limitarea rezultatelor la înregistrările fără corespondent în tabelul din dreapta

Dar dacă trebuie să efectuați uniunea la stânga inversă, adică să extrageți numai clienții care nu au o înregistrare corespondentă în tabelul din dreapta? Veți folosi tot o uniune la stânga pentru a vedea toate înregistrările din tabelul Client; pentru a obține rezultatul dorit, va trebui să adăugați criterii în clauza `Where`, ca să extrageți numai înregistrările al căror câmp ContactID este considerat nul. Numai înregistrările clienților care nu au înregistrări corespondente cu date de contact vor satisface aceste criterii. Codul următor îndeplinește această misiune:

```
SELECT organizationname
      From Client
        Left Outer Join On (contact.clientid = client.clientid)
      Where ClientType="B" And Contact.ContactId Is Null
      Order By organizationname
```

Figura 8.24 prezintă rezultatele acestei interogări.

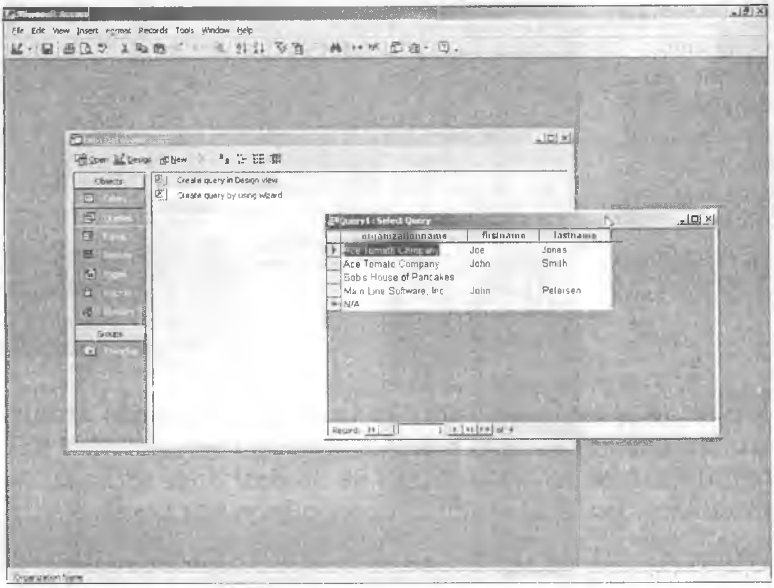


FIGURA 8.23
O uniune externă la stânga extrage toate înregistrările din partea stângă a interogării, indiferent dacă există înregistrări corespondente lor în tabelul din partea dreaptă.

Uniuni externe la dreapta și uniuni complet externe

Uniunile externe la dreapta și uniunile complet externe sunt discutate împreună în această carte deoarece este puțin probabil să le utilizați. Access nu suportă uniunile complet externe, dar pentru a realiza o prezentare completă vom discuta despre ambele tipuri de uniune în această secțiune.

Uniunile la stânga sunt utile deoarece permit existența unui părinte cu zero copii. O uniune la dreapta reprezintă scenariul invers, adică un copil fără părinte. Dacă vă amintiți discuția despre integritate referențială din capitolul 7, perspectiva existenței unei înregistrări copil fără o înregistrare părinte corespunzătoare era o violare a integrității referențiale.

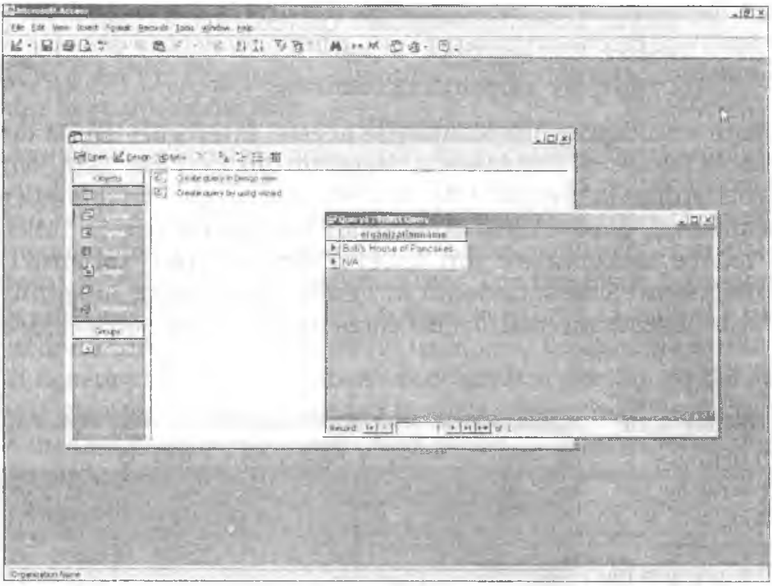


FIGURA 8.24
În plus față de realizarea uniunii la stânga, puteți și să extrageți numai acele înregistrări care nu au corespondent în tabelul din dreapta.

> **Notă** Primul lucru pe care îl veți învăța despre reguli este că, la un moment dat, veți întâlni situații în care acestea sunt încălcate. În viață, foarte puține reguli sunt imuabile. Pot exista situații când aveți o înregistrare copil fără înregistrare părinte. Gândiți-vă la situația unei facturi achitate cu numerar. Tot ce vă interesează este să vindeți ceva cuiva. Nu vă interesează cine este cumpărătorul (clientul) și nici nu aveți timp să înregistrați informațiile despre el. Vor fi alte situații când exact același sistem de baze de date trebuie să înregistreze informațiile despre client. În acest exemplu, vă confruntați cu o situație hibridă. O soluție simplă ar fi crearea unui client care plătește cu numerar și asocierea tututror facturilor achitate în numerar cu acest client. Deși soluția s-ar putea dovedi funcțională, acest pas suplimentar ar putea părea inacceptabil. Și atunci, poate că abordarea cea mai bună ar fi să permiteți existența unor facturi indiferent de existența înregistrărilor părinte. Problemele de proiectare a bazelor de date pentru astfel de situații depășesc însă scopul acestei cărți. Este suficient să știți că există situații când încălcarea regulilor are sens, iar limbajul SQL are capacitatea să rezolve aceste situații.

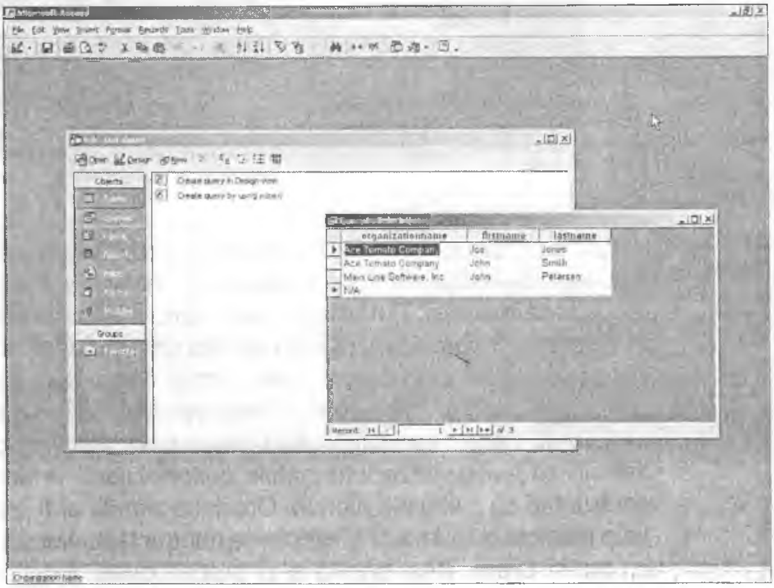
Înregistrările de acest tip se numesc orfane. Dacă într-o bază de date este impusă integritatea referențială, va fi imposibil să introduceți copii fără părinți. Totuși, există cazuri când relațiile părinte-copil sunt doar semantice. Cu alte cuvinte relația există doar cu numele dacă nu este impusă integritatea referențială. Codul următor prezintă utilizarea uniunii externe la dreapta:

```
SELECT organizationname, contact.firstname, contact.lastname
     From Client
```

```
Right Outer Join Contact On (contact.clientid=client.clientid)
Where ClientType = "B"
Order By organizationname
```

Figura 8.25 prezintă rezultatele acestei interogări. După cum vedeți firma Bob's House of Pancakes a dispărut din nou din listă. Nu uitați, scopul unei uniuni la dreapta este de a extrage înregistrări din tabelul din dreapta, indiferent dacă există înregistrări corespondente în tabelul din stânga. Fiecare înregistrare din tabelul din dreapta, tabelul Contact, are o înregistrare corespondentă în tabelul din stânga, tabelul Client. Nici una dintre aceste înregistrări nu corespunde firmei Bob's House of Pancakes.

FIGURA 8.25
O uniune externă la dreapta selectează înregistrările din tabelul copil, indiferent dacă există înregistrări corespunzătoare în tabelul părinte.



O uniune complet externă combină uniunile externe la dreapta și la stânga. Sunt selectate toate datele, din ambele tabele, indiferent dacă există date corespondente lor în tabelul celălalt. Deși multe baze de date, ca de exemplu SQL Server, Oracle și DB2 suportă uniunile complet externe, Access nu suportă acest tip de uniune. Dar chiar dacă Access ar fi suportat acest tip de uniune, situațiile în care sunt necesare uniunile complet externe ar fi relativ rare.

Combinarea mai multor seturi de rezultate: utilizarea clauzei Union

Combinarea mai multor seturi de rezultate este un subiect destul de avansat. Totuși, efectul clauzei `Union` este ușor de înțeles. Clauza `Union` vă permite

să combinați două declarații `Select` într-un singur set de rezultate și veți avea nevoie de această clauză de multe ori. Considerați situația în care trebuie să realizați o listă consolidată a clienților. La început, treaba poate părea simplă, dar în realitate nu este.

Declarația `Select` a fost prezentată anterior în acest capitol, iar exemplele de linii de cod au fost prezentate tehnicile care vă permit să extrageți date despre clienți din tabelul Client. În cazul clienților persoane fizice, numele clientului este memorat în câmpurile firstname și lastname. În cazul firmelor, numele este stocat în câmpul organizationname. Crearea unui extras cu toate aceste trei câmpuri nu este dificilă. Dar dacă vi se cere să realizați un extras cu numai două câmpuri? Primul câmp trebuie să fie tipul clientului, care arată dacă acesta este persoană fizică sau juridică (firmă). Al diolea câmp se numește *clientname* și memorează numele clientului. După cum știți, nu există un câmp cu acest nume. Știți și că, utilizând clauza As, puteți redenumi ușor un câmp din setul de rezultate. Totuși, vă revine sarcina de a reuni datele din câmpurile firstname și lastname, precum și cele din câmpul orgenizationname într-un singur câmp cu numele clientului. Acum, când știți datele problemei, să trecem la rezolvarea ei. Următor bloc de instrucțiuni duce la rezultatele dorite:

```
SELECT clienttype, lastname + ", " + firstname As client
      From Client
      Where ClientType = "I"
Union
SELECT clienttype, organizationname As Client
      From Client
      Where ClientType = "B"
      Order By ClientType
```

Figura 8.26 prezintă rezultatele acestei interogări.

Necesitatea clauzei Union este dată de faptul că toate rezultatele selectate trebuie să aibă același număr de câmpuri, iar datele din același câmp trebuie să aibă același tip. Nu uitați, există un singur set de rezultate. Prima interogare va defini structura setului de rezultate, iar după acest moment toate înregistrările selectate trebuie să aibă același număr de câmpuri. În plus, datele aceluiași câmp trebuie să fie toate de același tip. În acest caz, deși al doilea câmp diferă în cele două interogări prin faptul că în prima sunt selectate numele și prenumele clienților persoane fizice, iar în a doua sunt selectate numele firmelor, ambele sunt câmpuri de tip text. În final, întrucât prima interogare dictează structura setului de rezultate, nu mai este necesară clauza As după câmpul organizationname în a doua interogare. De fapt, a doua clauză As este ignorată. Totuși, pentru consecvență, este o idee bună să etichetați corespunzător câmpurile.

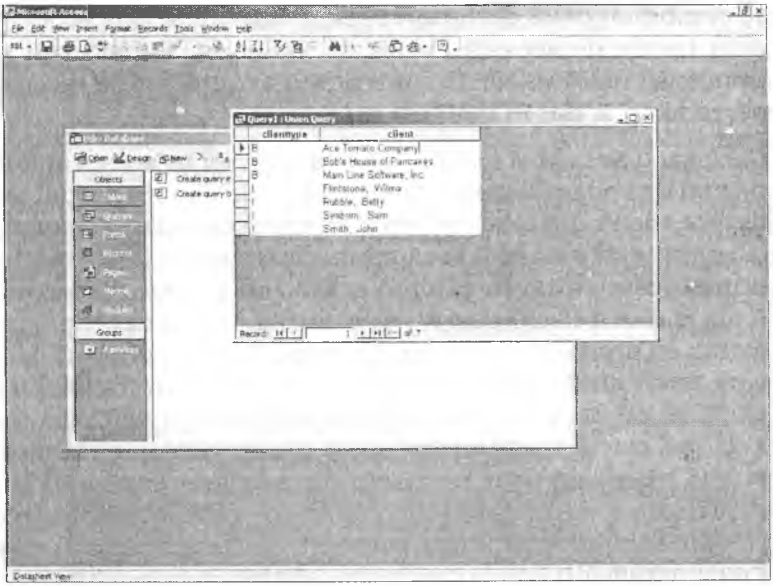


FIGURA 8.26
Clauza `Union` vă permite să combinați într-un singur set de rezultate datele din mai multe declarații `Select`.

Rezumat

Acest capitol v-a oferit o prezentare generală a limbajului SQL și a uneia dintre cele mai utilizate părți ale acestui limbaj, declarația `Select`. Ați învățat cum să utilizați comanda SQL `Select` pentru a extrage date din tabelele unei baze de date. Clauza `From` specifică tabelul de bază care suportă interogarea. Clauza `Where` oferă un mecanism pentru filtrarea datelor din setul de rezultate. Interogările de tip `Top n` și clauza `Distinct` reprezintă alte instrumente pentru a restrânge datele din setul de rezultate. Clauza `Join` vă permite să extrageți date din mai multe tabele. Pe lângă toate acestea, ați învățat și să folosiți clauza `Union` pentru a combina datele din mai multe interogări într-un singur set de rezultate. În plus, ați folosit clauza `As` pentru a specifica numele câmpului din setul de rezultate. Această clauză este foarte utilă mai ales când combinați datele din mai multe câmpuri într-un singur câmp din setul de rezultate. După ce ați înțeles elementele de bază ale limbajului SQL, puteți trece la niște funcții SQL mai avansate.

9

SQL – DINCOLO DE GENERALITĂȚI

În acest capitol:

- A doua organizare a rezultatelor: utilizarea clauzei `Group By`
- Utilizarea funcțiilor avansate
- A doua filtrare a rezultatelor: utilizarea clauzei `Having`
- A treia filtrare a rezultatelor: crearea declarațiilor SQL parametrizate
- Modificarea datelor folosind declarațiile `Insert`, `Update` și `Delete`
- Rezumat

În capitolul anterior, am prezentat o introducere în unul dintre cele mai puternice programe existente – SQL. SQL este limbajul bazelor de date. Dacă aveți de gând să dezvoltați baze de date, va trebui să învățați și să înțelegeți acest program. În capitolul anterior ați învățat să selectați și să organizați datele din mai multe tabele. În acest capitol, veți face un pas înainte, învățând să agregați (totalizați) și să sumarizați datele. În capitolul anterior, ați învățat să realizați o listă a clienților, organizați pe tipuri. Dar dacă trebuie să realizați o interogare care produce o listă cu tipurile clienților și numărul de clienți de fiecare tip? Acest tip de interogare vă cere să calculați numărul de clienți de fiecare tip. SQL poate rezolva și această problemă. Acest tip de operații de agregare, ca și operațiile de manipulare a datelor cum sunt includerea, actuallizarea și ștergerea datelor reprezintă scopul acestui capitol.

A doua organizare a rezultatelor: utilizarea clauzei `Group By`

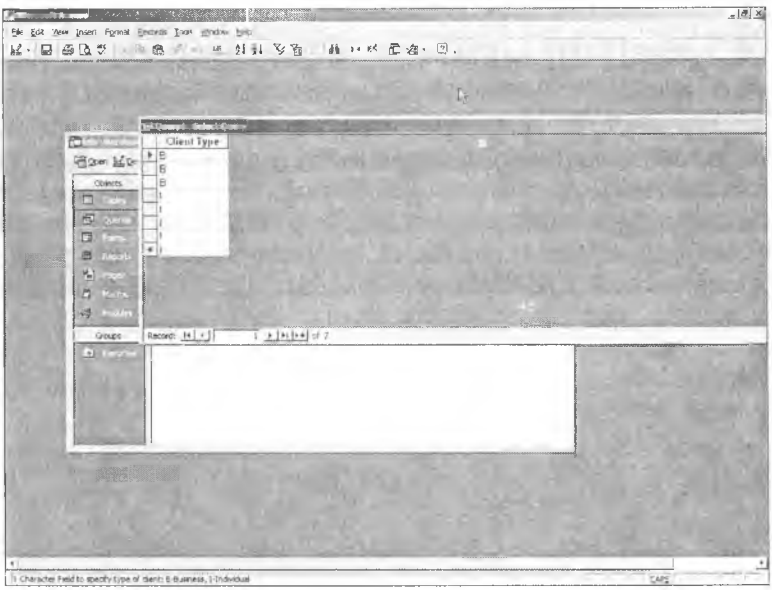
Primul pas pentru agreagarea datelor presupune utilizarea clauzei `Group By`. Aceasta este un mecanism de sumarizare a datelor. Pentru exemplificare, să considerăm interogarea:

```
Select clienttype
      From Client
      Order By clienttype
```

Figura 9.1 prezintă rezultatele acestei interogări.

FIGURA 9.1

O listă simplă cu tipurile de clienți care apar în tabelul Client.



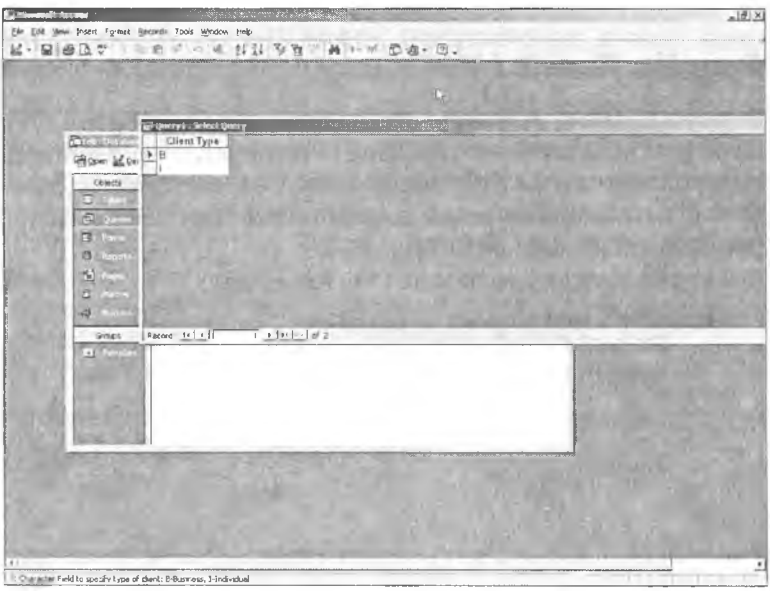
Aceste rezultate nu sunt teribil de utile, nu-i așa? Totuși, există unele informații utile care pot fi observate fără prea mult efort. Din aceste rezultate, puteți vedea că există numai două tipuri de clienți – B și I (firme și persoane fizice). Și dacă veți aloca ceva timp pentru a număra manual diferitele înregistrări, veți vedea că printre clienți se află trei firme și patru persoane fizice. Deși este bine să dețineți aceste informații, nu este nevoie de un efort prea mare pentru a le obține. Unul dintre avantajele pe care le oferă o bază de date este faptul că puteți organiza datele pentru a obține rapid astfel de informații. Valoarea datelor depinde de timp. Cu cât datele sunt mai vechi, cu atât valoarea lor este mai mică. Dacă cineva dorește să totalizeze și să sumarizeze datele manual, probabil că atunci când va termina, datele vor fi depășite. Și există întotdeauna riscul erorilor de numărare. Aici intervine clauza `Group By`.

Clauza Group By vă permite să sumarizați datele. Această operațiune este primul pas pentru agregarea și subtotalizarea datelor. Următoarea interogare prezintă clauza Group By, iar figura 9.2 prezintă rezultatele ei:

```
Select clienttype
      From client
      Group By clienttype
      Order By clienttype
```

FIGURA 9.2

Clauza `Group By` oferă posibilitatea de a sumariza datele.



Interogarea precedentă este echivalentă din punct de vedere funcțional cu următoarea:

```
Select Distinct clienttype
      From Client
      Order By clienttype
```

De aceea, pentru a organiza clar înregistrările, clauza `Group By` este a doua metodă de sumarizare a datelor pe care ați învățat-o în SQL. Diferența esențială între clauzele `Distinct` și `Group By` este că ultima vă permite să agregați și să subtotalizați datele. Următoarea secțiune are ca subiect agregarea datelor.

Utilizarea funcțiilor de agregare

Vor fi foarte rare cazurile când veți utiliza clauza `Group By` fără a folosi una dintre funcțiile de agregare din SQL. Cele cinci funcții de agregare din SQL sunt `Count()`, `Sum()`, `Avg()`, `Min()` și `Max()`. Numele lor descriu destul de bine ceea ce fac. De exemplu, funcția `Count()` totalizează numărul de înregistrări, funcția `Sum()` calculează totalul matematic al valorilor câmpurilor, iar `Avg()` calculează media aritmetică a valorilor. În următoarele trei secțiuni vor fi prezentate detalii despre funcționarea funcțiilor de agregare.

Funcția `Count()`

Funcția `Count()` numără înregistrările din fiecare grup. În acest caz, există două grupuri, unul pentru firme și unul pentru persoane fizice. Întrebarea la care încercați să răspundeți este „Câți clienți de fiecare tip are firma?". Codul următor ilustrează funcționarea funcției `Count()`, iar figura 9.3 prezintă rezultatele obținute:

```
Select clienttype,count(*) As count
      From client
      Group By clienttype
      Order By Count(*) Desc
```

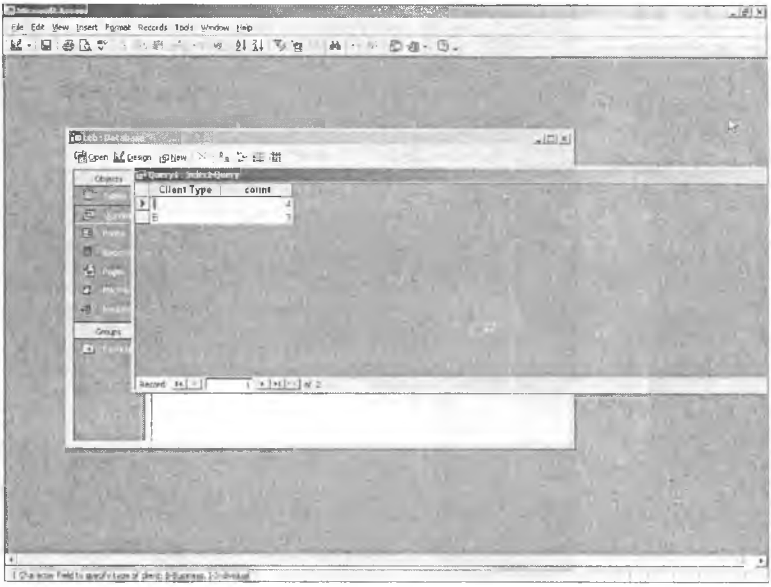


FIGURA 9.3

Funcția de agregare `Count()` vă permite să determinați numărul de înregistrări din fiecare grup.

Oricând veți folosi una dintre funcțiile de agregare din SQL, veți utiliza clauza As pentru a specifica numele coloanei în setul de rezultate. Altfel, este posibil ca motorul bazei de date să-i aloce un nume care nu este atât de descriptiv precum cel pe care l-ați alege dumneavoastră. În plus, rezultatele sunt numerele de înregistrări, afișate în ordine descrescătoare.

Ținând cont și de ceea ce ați învățat în capitolul anterior, poate veți dori să filtrați setul de rezultate folosind clauza `Where`. Codul următor este identic celui din exemplul precedent, cu o singură excepție. În acest caz, sunt numărați doar clienții din statul California:

```
Select clienttype,count(*) As count
      From client
      Where state = 'CA'
      Group By clienttype
      Order By Count(*) Desc
```

Figura 9.4 prezintă rezultatele care apar. După cum vedeți, există doi clienți din statul California, ambii fiind persoane fizice.

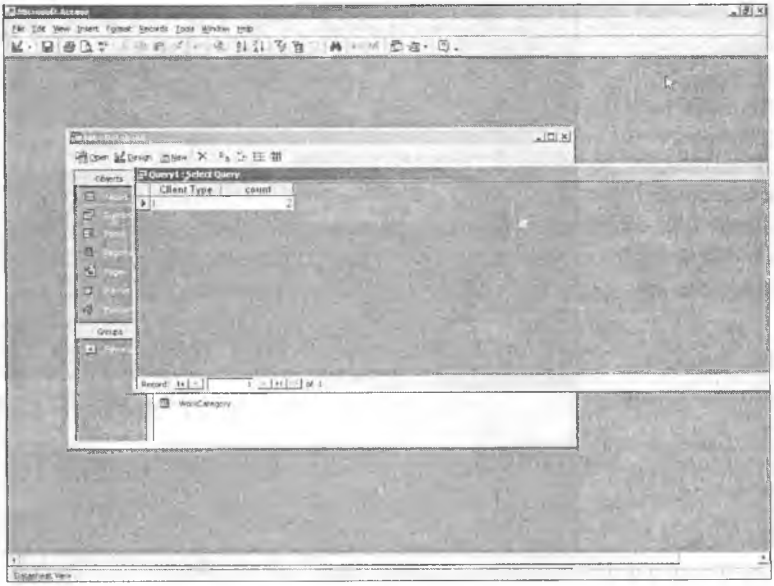


FIGURA 9.4

În combinație cu clauza `Where`, puteți să agregați și să filtrați datele în același timp.

Operațiuni avansate: metodologia prin care datele pot fi puse la un loc

Să ne oprim câteva clipe pentru a examina puterea limbajului SQL prin intermediul unui exemplu mai complicat. Vi s-a cerut să extrageți din baza de date informațiile următoare:

- Lista cazurilor și a angajaților care lucrează la ele
- Lista cazurilor și numărul de angajați care lucrează la ele
- Lista cazurilor și numărul de tipuri de angajați care lucrează la ele

Oricând vă confruntați cu o solicitare de extragere a datelor, primul pas ar trebui să fie identificarea tabelelor necesare. Examinând cele trei cerințe, observați că sunt necesare cel puțin trei tabele:

- `Case`
- `Employee`
- `EmployeeClass`

Dacă vă amintiți, nu există o relație directă între tabelele `Case` și `Employee`. La un caz pot lucra mai mulți angajați, iar un angajat poate lucra la mai multe cazuri. A fost necesar un tabel care rezolvă relația mulți la mulți, numit `EmployeeCase`. De aceea, va fi nevoie de un al patrulea tabel. Figura 9.5 prezintă o parte din diagrama bazei de date TEB care conține tabelele necesare și relațiile dintre ele.

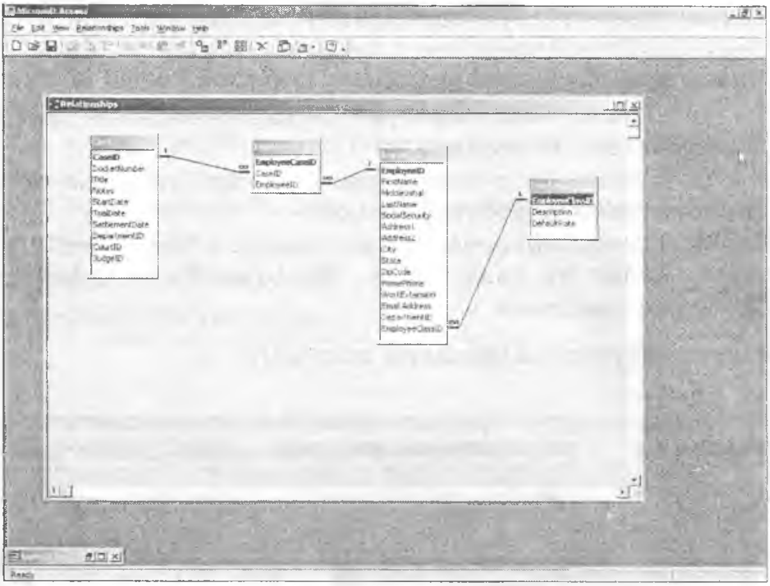


FIGURA 9.5
Identificarea tabelelor necesare și a relațiilor dintre acestea este primul pas în construirea unei declarații SQL.

Cu figura 9.5 în minte, puteți determina clauzele `From` și `Join`:

```
FROM (EmployeeClass INNER JOIN Employee ON
        EmployeeClass.EmployeeClassID = Employee.EmployeeClassID)
      INNER JOIN (Case INNER JOIN EmployeeCase ON Case.CaseID =
EmployeeCase.CaseID)
      ON Employee.EmployeeID = EmployeeCase.EmployeeID
```

Pasul următor presupune determinarea câmpurilor de date care trebuie returnate în setul de rezultate. Pe baza informațiilor solicitate, vor fi necesare următoarele câmpuri:

- `case.title`
- `employee.firstname` și `employee.lastname`
- `employeeclass.description`

În afara câmpurilor specificate, va trebuie să mai determinați și:

- Numărul de angajați
- Numărul tipurilor de angajați

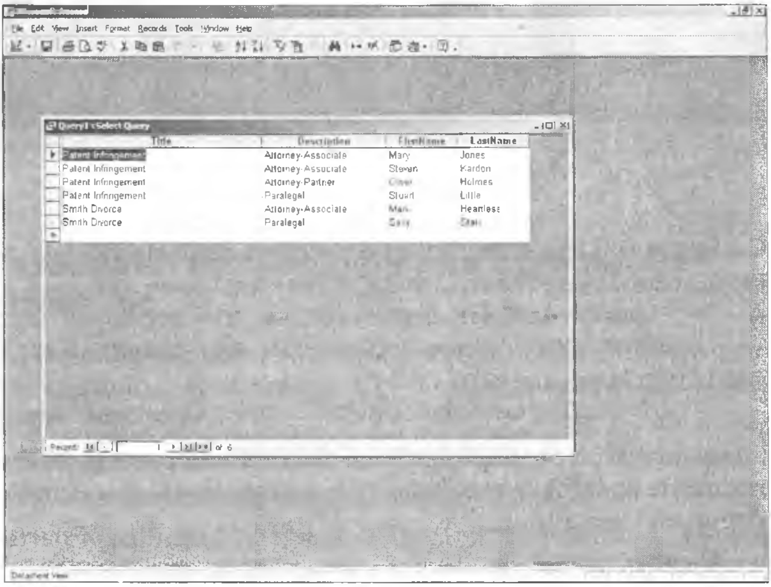
Să abordăm mai întâi prima interogare: lista cazurilor și a angajaților care lucrează la ele. Această interogare va servi ca fundație pentru interogările următoare, care vor agrega și sumariza datele. Codul următor va furniza lista cerută:

```
Select Case.Title, EmployeeClass.Description,
Employee.FirstName, Employee.LastName
      From (EmployeeClass
        Inner Join Employee ON EmployeeClass.EmployeeClassID =
Employee.EmployeeClassID)
        Inner Join(Case Inner Join EmployeeCase On Case.CaseID =
EmployeeCase.EmployeeCaseID)
On Employee.EmployeeID=EmployeeCase.EmployeeID
      Order By Case.Title, EmployeeClass.Description,
Employee.Lastname;
```

Figura 9.6 prezintă rezultatul interogării.

FIGURA 9.6

Setul de rezultate conține date din tabelele `Case`, `EmployeeClass` și `Employee`.



Este important să observați că singurul scop al tabelului `EmloyeeCase` este să faciliteze uniunea între tabelele `Employee` și `Case`. Dacă nu ar fi existat acest tabel intermediar, nu ar fi fost posibil să asociați mai multe cazuri cu mai mulți angajați.

Având la dispoziție interogarea de bază, să trecem la interogarea următoare – o listă cu cazurile și numărul de angajați care lucrează la ele. Întrucât nu vă preocupă numele acestor angajați, tabelul `Employee` poate fi exclus din interogare. Dacă singura informație de interes este numărul angajaților, aceasta va fi furnizată de tabelul `EmployeeCase`. În plus, în acest caz puteți renunța și la tabelul `EmployeeClass`.

Dacă agregați datele, automat înseamnă că este necesară clauza `Group By`. Pentru această interogare este necesară o listă cu cazurile și numărul angajaților care lucrează la fiecare caz. Deci rezultatele trebuie să fie grupate după cazuri, iar următorul cod îndeplinește această funcție:

```
Select Case.Title, Count(employeecase.employeecaseid) As employees
      From Case
        Inner Join EmployeeCase On Case.CaseID = EmployeeCase.CaseID
      Group By case.title
      Order By Case.Title
```

Figura 9.7 prezintă rezultatele care apar.

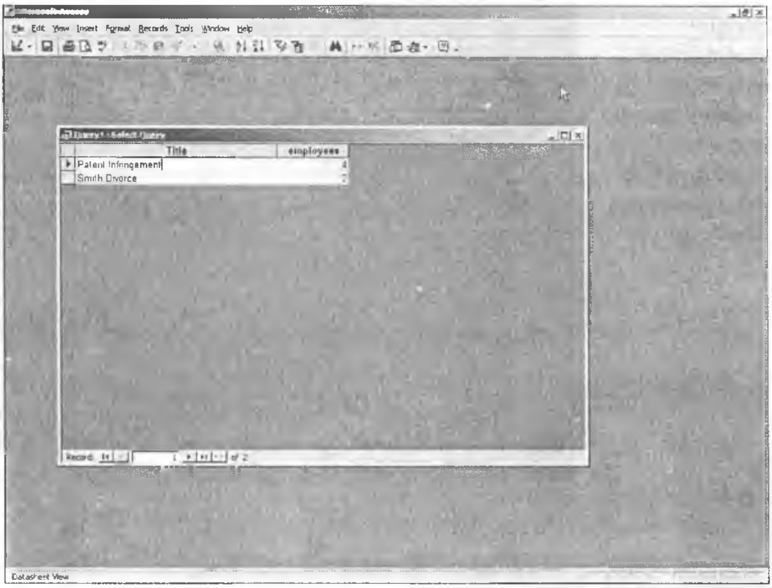


FIGURA 9.7
Acest set de rezultate conține numărul de angajați grupați pe cazuri.

Următoarea interogare o ia pe cea precedentă și mai parcurge un nivel. În interogarea precedentă ați fost interesat de numărul angajaților. În această interogare, veți număra tipurile de angajați din totalul angajaților care lucrează la fiecare caz. De exemplu, la cazul nerespectării dreptului de proprietate asupra patentului, lucrează patru angajați. Care este distribuția celor patru angajați în acest caz? Aceasta este întrebarea la care veți răspunde aici. Codul următor îndeplinește această funcție:

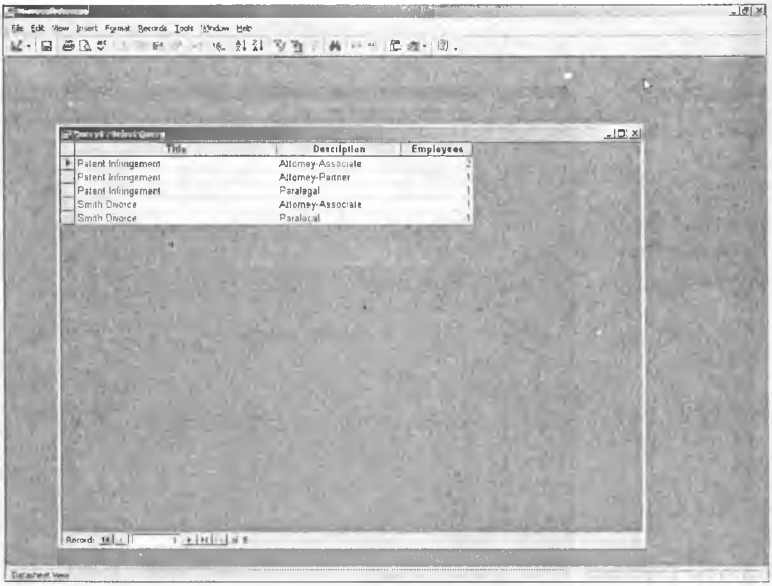
```
Select Case.Title, EmployeeClass.Description,
Count(EmployeeCase.Employeeid) As Employees
```

```
      From (EmployeeClass
        Inner Join Employee ON EmployeeClass.EmployeeClassID =
Employee.EmployeeClassID)
        Inner Join(Case Inner Join EmployeeCase On Case.CaseID =
EmployeeCase.CaseID) On Employee.EmployeeID = EmployeeCase.EmployeeID
      Group By Case.Title,EmployeeClass.Description
      Order By Case.Title,EmployeeClass.Description
```

Figura 9.8 prezintă rezultatele care apar.

FIGURA 9.8

Setul de rezultate conține numărul de tipuri de angajați grupați după `case.title` și `employeeclass.description`.



Pe măsură ce „forați" mai adânc în date, veți avea mai multe câmpuri în clauza `Group By`. În această interogare, deși nu sunt returnate numele angajaților, tabelul `Employee` a fost necesar pentru a returna tipul angajaților. Cu alte cuvinte, pentru a trece de la tabelul `Case` la tabelul `EmployeeClass`, sunt necesare tabelele `EmployeeCase` și `Employee` pentru a facilita uniunea.

Funcția `Sum()`

Deseori, va trebui să adunați datele din diferite înregistrări în loc să numărați înregistrările. Aici intervine funcția de agregare `Sum()`. Din fericire, ați rezolvat deja o mare parte din problemele dificile în secțiunea anterioară. Aceleași concepte de la funcția `Count()` se aplică și pentru funcția `Sum()`.

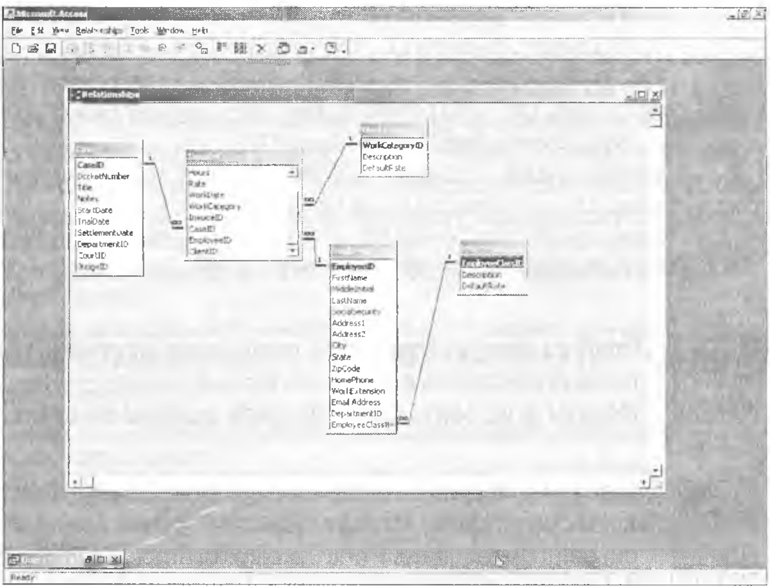
Pentru a ilustra funcționarea funcției `Sum()`, să considerăm următoarele cerințe:

- Lista cazurilor, angajaților, orelor și tarifelor aplicate;
- Lista cazurilor, totalul orelor și tarifelor aplicate grupate pe angajați;
- Lista cazurilor, totalul orelor și tarifelor aplicate grupate pe tipuri de angajați;
- Lista cazurilor, totalul orelor și tarifelor aplicate grupate pe cazuri;
- Lista cazurilor, totalul orelor și tarifelor aplicate grupate pe categorii de activități;

Aplicând metodologia învățată în secțiunea precedentă, primul pas este identificarea tabelelor necesare pentru a răspunde acestor cerințe. De la început, știți că aveți tabelele `Employee` și `Case`. Poate credeți că este necesar și tabelul `EmployeeCase`. După cum veți vedea, tabelul `EmployeeCase` nu este necesar pentru aceste interogări. În plus va trebui să mai utilizați și tabelele `EmployeeClass` și `WorkCategory`. În final, pentru a obține orele și tarifele, vă este necesar tabelul `TimeEntryDetail`. Tabelul `TimeEntryDetail` se ocupă de uniunea între tabelele `Employee` și `Case`. Figura 9.9 prezintă partea relevantă pentru acest set de interogări a diagramei bazei de date TEB.

FIGURA 9.9

Tabelele necesare pentru a extrage informații despre orele de muncă și tarifele aplicate, grupate după angajați, cazuri, categorii de activități și clase de angajați.



Uniunile prezentate în figura 9.9 sunt facilitate de următorul cod SQL:

```
    FROM WorkCategory INNER JOIN ((EmployeeClass INNER JOIN
Employee ON EmployeeClass.EmployeeClassID = Employee.EmployeeClassID)
```

```
        INNER JOIN([Case] INNER JOIN TimeEntryDetail ON
Case.CaseID = TimeEntryDetail.CaseID) ON Employee.EmployeeID
= TimeEntryDetail.EmployeeID) ON
        WorkCategory.WorkCategoryID = TimeEntryDetail.WorkCategory
```

Notă

Până acum, ați învățat că, pe măsură ce crește numărul de tabele necesare pentru o interogare, cu atât este mai complexă clauza Join. Nu vă faceți probleme! În capitolul următor, veți vedea cum pot fi create grafic înregistrările. Aplicația Access Query Designer oferă o interfață prietenoasă față de utilizator. În culise, este construit pentru dumneavoastră codul SQL. Scopul acestui capitol și al celui precedent este prezentarea limbajului SQL, independent de o anumită aplicație de proiectare a interogărilor. Nu uitați, este posibil să vă aflați într-un mediu în care trebuie să creați manual declarațiile SQL!

Prima interogare pe care trebuie să o creați este lista brută a datelor care vor fi agregate și sumarizate prin interogările următoare. Aceste date sunt furnizate de codul SQL următor:

```
SELECT employee.firstname + ' ' + employee.lastname As employee,
        EmployeeClass.Description As employeeclass,
        Case.Title, WorkCategory.Description AS workcategory,
        TimeEntryDetail.Description AS workdescription,
        TimeEntryDetail.Hours, TimeEntryDetail.Rate,
        (timeentrydetail.hours*timeentrydetail.rate) AS total
      FROM WorkCategory INNER JOIN ((EmployeeClass INNER
JOIN Employee ON EmployeeClass.EmployeeClassID =
        Employee.EmployeeClassID)
      INNER JOIN ([Case] INNER JOIN TimeEntryDetail ON
Case.Case.ID = TimeEntryDetail.CaseID) ON
      Employee.EmployeeID = TimeEntryDetail.EmployeeID) ON
      WorkCategory.WorkCategoryID = TimeEntryDetail.WorkCategory
```

Indiciu

Priviți cu atenție câmpul total. Interogarea returnează câmpurile cu numărul de ore și tarifele aplicate din tabelul `TimeEntryDetail`. În plus, este returnat și un câmp care înmulțește numărul de ore cu tariful aplicat.

Indiciu

Clauza `As` vă ajută să evitați ambiguitatea. Câmpul Description din tabelele `EmployeeClass`, `WorkCategory` și `Time EntryDetail` face parte din setul de rezultate. Pentru a elimina ambiguitățile, este utilizată clauza `As` pentru a da câmpului din setul de rezultate un nume mai descriptiv.

Figura 9.10 prezintă setul de rezultate neprelucrat.

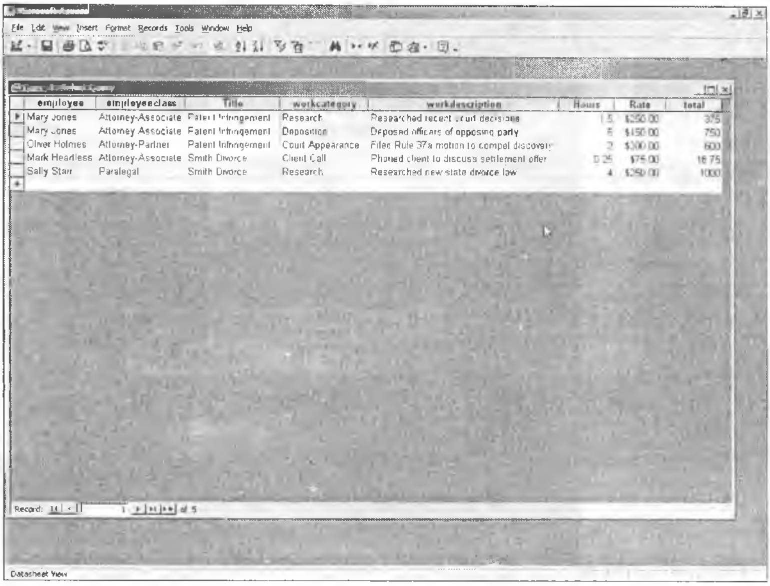


FIGURA 9.10
Lista brută a datelor sumarizează grupurile de angajați, cazuri, tipuri de activități și clase de angajați.

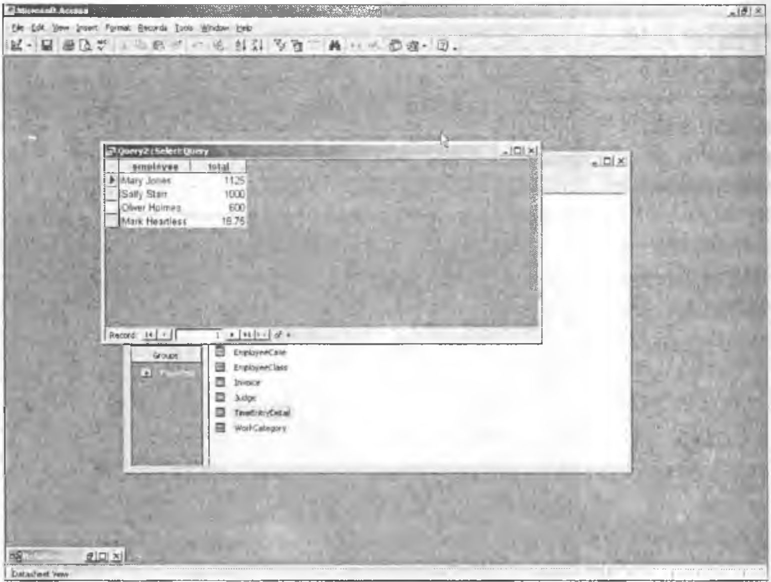
Este important să observați că nu sunt necesare toate tabelele pentru fiecare interogare. Când intrați în detalii, atunci veți avea nevoie de mai multe tabele pentru a suporta un nivel mai mare de granularitate a detaliilor. Figura 9.10 prezintă cea mai mare cantitate de detalii. Acum veți începe călătoria de 10000 de pași (aluzie la proverbul chinez *O călătorie de 10000 de pași începe cu primul pas.* – n. trad.) și veți lucra independent.

Prima interogare presupune sumarizarea datelor după angajați. Codul următor rezolvă acest lucru:

```
SELECT employee.firstname+' '+employee.lastname AS employee,
       Sum(timeentrydetail.hours*timeentrydetail.rate) AS total
      FROM Employee
        INNER JOIN TimeEntryDetail ON Employee.EmployeeID =
TimeEntryDetail.EmployeeID
        GROUP BY employee.firstname+' '+employee.lastname
       Order By Sum(timeentrydetail.hours*timeentrydetail.rate) Desc
```

Figura 9.11 prezintă setul de rezultate ale interogării.

FIGURA 9.11
Acest set de rezultate totalizează numărul de ore înmulțit cu tariful aplicat și sumarizează după angajați.



Ce ați zice de sumarizare pe baza categoriei de activitate? Categoriile de activități există în două contexte. Mai întâi, ele există în mod independent. Apoi, ele există în funcție de angajați. Cu alte cuvinte, puteți sumariza datele numai pe baza categoriilor de activități sau pe baza categoriilor de activități și de angajați. Codul următor sumarizează datele pe baza a două categorii:

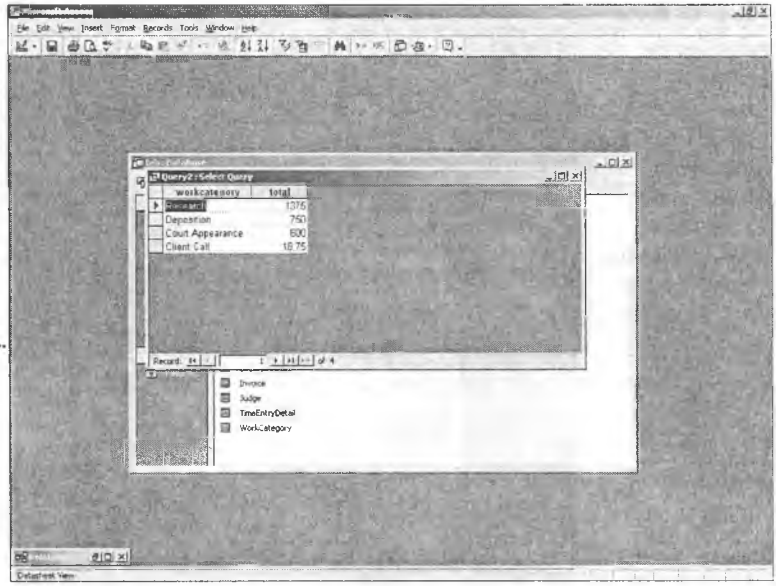
```
SELECT WorkCategory.Description AS workcategory,
        Sum(timeentrydetail.hours*timeentrydetail.rate) AS total
       FROM WorkCategory
        INNER JOIN TimeEntryDetail ON WorkCategory.WorkCategoryID =
TimeEntryDetail.WorkCategory
        GROUP BY WorkCategory.Description
        ORDER BY Sum(timeentrydetail.hours*timeentrydetail.rate) Desc
```

Figura 9.12 prezintă rezultatele acestei interogări.

Figurile 9.11 și 9.12 prezintă două puncte de vedere asupra acelorași date. Figura 9.11 prezintă datele sumarizate după angajați, iar figura 9.12 prezintă datele sumarizate după categoria de activitate. Capacitatea de a diseca, răsuci și sumariza datele în acest mod este acea valoare pe care o furnizează organizației limbajul SQL și baza de date. Ce ați zice de o combinație între aceste două interogări – un sumar al activității după categoriile de activități și după angajați? Acest lucru necesită două nivele de grupare. Așa cum am menționat deja, cu cât sunt furnizate mai multe detalii, cu atât mai multe tabele vor fi necesare în interogare. Codul următor realizează această interogare mai detaliată:

FIGURA 9.12

Setul de rezultate totalizează numărul de ore înmulțit cu tariful și sumarizează datele după categoriile de activități.




```
SELECT WorkCategory.Description AS workcategory,
        employee.firstname+' '+employee.lastname AS employee,
        Sum(timeentrydetail.hours*timeentrydetail.rate) AS total
      FROM WorkCategory
        INNER JOIN (Employee INNER JOIN TimeEntryDetail ON
Employee.EmployeeID =
        TimeEntryDetail.EmployeeID)
        ON WorkCategory.WorkCategoryID = TimeEntryDetail.WorkCategory
      GROUP BY workcategory.description,employee.firstname+'
'+employee.lastname
      ORDER BY workcategory.description,Sum
(timeentrydetail.hours*timeentrydetail.rate) Desc
```

Figura 9.13 prezintă rezultatele acestei interogări.

Să ne oprim puțin pentru a diseca o parte din date. Figura 9.12 arată că activitățile legate de cercetare au fost taxate cu 1375 $. Figura 9.13 arată că Sally Star a generat 1000 $, iar Mary Jones a generat 375 $ prin activități de cercetare. Dacă privim figura 9.11, vedem că Mary Jones a generat un venit de 1125 $. Cum au fost distribuiți ceilalți 750 $ pe care Mary i-a câștigat? Figura 9.13 arată că restul sumei a fost obținut prin activități de luare a depozițiilor.

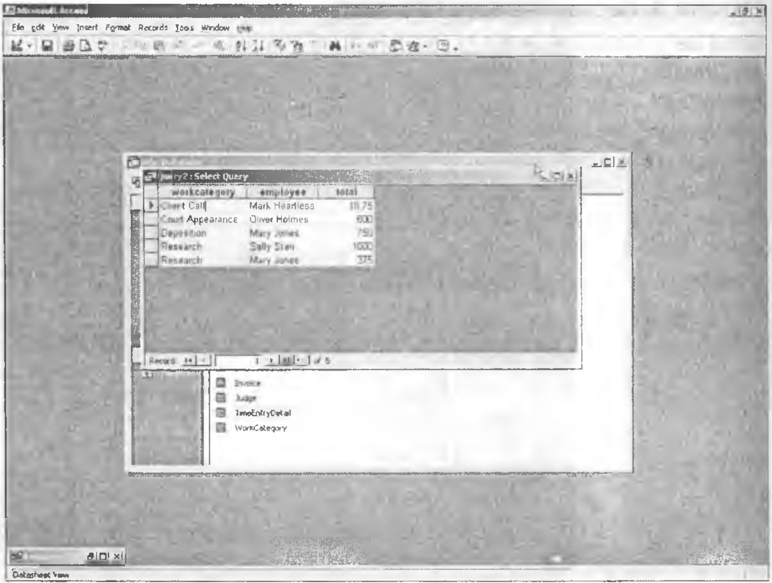


FIGURA 9.13
Acest set de rezultate totalizează numărul de ore înmulțit cu tariful și le sumarizează după categoriile de activități și după angajați.

Până acum, ar fi trebuit să observați că există un model care se repetă. Nivelul de detaliere de care aveți nevoie este direct legat de numărul de tabele necesare, de numărul câmpurilor din setul de rezultate și de conținutul clauzei `Group By`. Pentru a furniza un argument și mai solid, să considerăm un set de rezultate care sumarizează după caz. Codul următor este foarte asemănător celui precedent care că sumarizează după categoria de activitate și cu cel care sumarizează după angajați:

```
SELECT Case.Title,
       Sum(timeentrydetail.hours*timeentrydetail.rate) AS total
     FROM Case
       INNER JOIN TimeEntryDetail ON Case.CaseID =
TimeEntryDetail.CaseID
       GROUP BY Case.Title
      ORDER BY Sum(timeentrydetail.hours*timeentrydetail.rate) Desc
```

Figura 9.14 prezintă rezultatele acestei interogări.

Acum, vi se pare complicat să sumarizați mai departe această interogare după angajați și după categoriile de activități? Procesul nu este complicat: trebuie numai să introduceți tabelul corespunzător și să realizați uniunea cu interogarea, să introduceți câmpurile suplimentare la lista de selecție și, în final, să introduceți câmpurile în clauzele `Order By` și `Group By`.

Dar dacă doriți numai să determinați totalul general al veniturilor generate? Deoarece nu sunt necesare detalii, vă așteptați ca declarația SQL să fie foarte simplă. Iată codul – judecați și dumneavoastră!

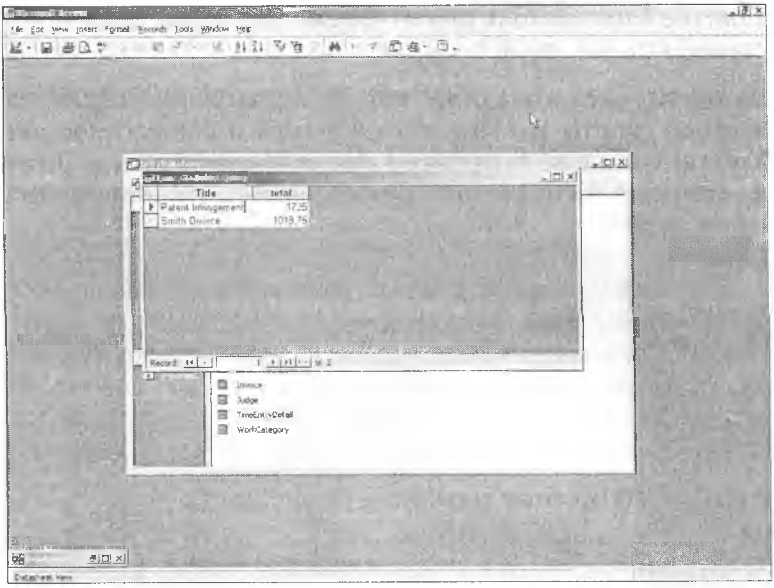


FIGURA 9.14
Setul de rezultate totalizează numărul de ore înmulțit cu tariful și sumarizează datele după cazuri.

```
SELECT Sum(timeentrydetail.hours*timeentrydetail.rate) AS total
    FROM TimeEntryDetail
```

Figura 9.15 prezintă rezultatele acestei interogări.

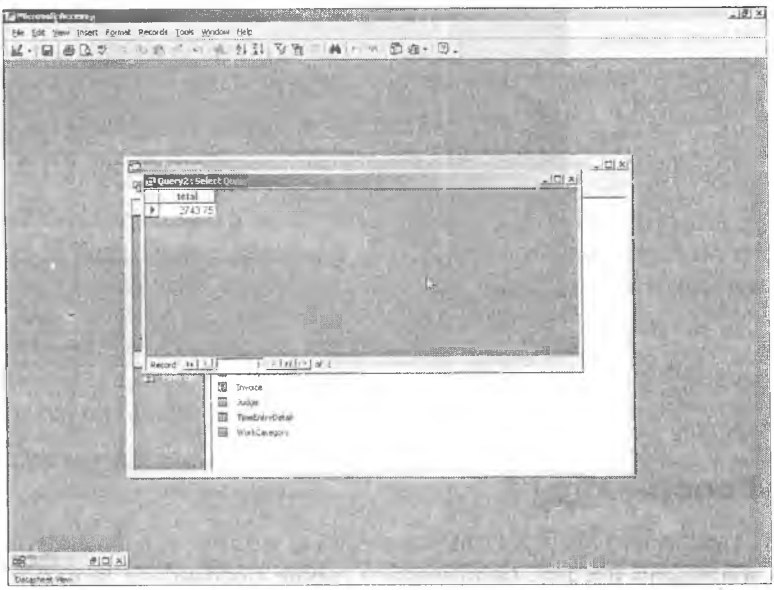


FIGURA 9.15
Setul de rezultate furnizează totalul general al numărului de ore înmulțit cu tariful.

Doarece este calculat totalul datelor din întreg tabelul, nu sunt necesare clauzele `Order By` și `Group By`.

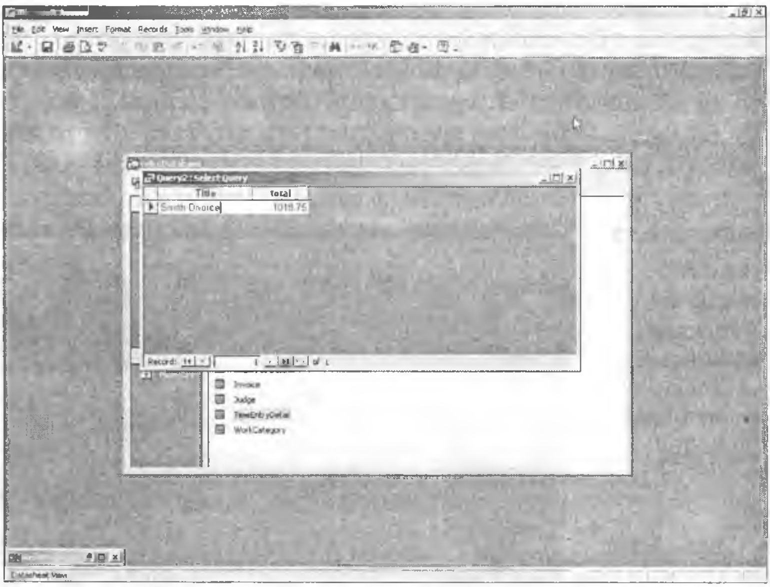
În sfârșit, dacă erați preocupat de sumarizarea datelor pentru un singur angajat, pentru un singur caz, pentru o categorie de activități ș.am.d.? Trebuie numai să introduceți clauza `Where` pentru a filtra datele din setul de rezultate. Codul următor extrage datele de sumarizare pentru cazul de divorț Smith:

```
SELECT Case.Title,
     Sum(timeentrydetail.hours*timeentrydetail.rate) AS total
    FROM Case
     INNER JOIN TimeEntryDetail ON Case.CaseID=TimeEntryDetail.Caseid
    WHERE case.title="Smith Divorce"
    GROUP BY Case.Title
    ORDER BY Sum(timeentrydetail.hours*timeentrydetail.rate) Desc
```

Figura 9.16 prezintă rezultatele acestei interogări.

FIGURA 9.16

Setul de rezultate furnizează un total al veniturilor obținute într-un anumit caz folosind clauza `Where`.



Funcția `Avg()`

Funcția `Avg()`, așa cum sugerează și numele ei (*avg* este prescurtarea de la *average* = medie), calculează o medie pornind de la expresia introdusă în argumentul ei. Codul următor este un exemplu simplu de utilizare a acestei funcții:

```
SELECT Avg(defaultrate) As AverageRate
      From Employeeclass
```

Figura 9.17 prezintă rezultatele acestei interogări.

FIGURA 9.17
Funcția `Avg()` calculează media expresiei din argument.

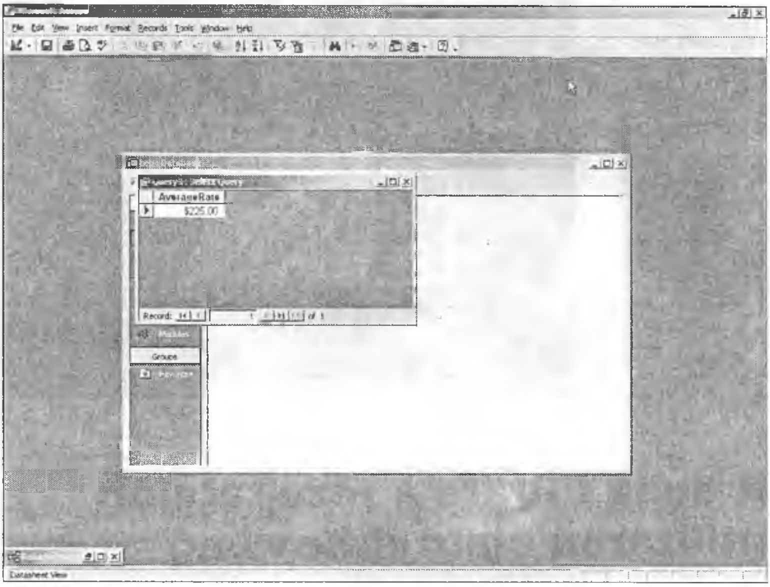


Figura 9.17 prezintă tariful mediu aplicat de tipurile de angajați ai firmei.

Funcțiile `Min()` și `Max()`

Dar dacă scopul dumneavoastră ar fi determinarea valorilor minimă și maximă ale unui anumit câmp dintr-un tabel? Aici intervin funcțiile de agregare `Min()` și `Max()`. Codul următor ilustrează funcționarea acestor funcții:

```
Select Min(defaultrate) As minrate,
        Max(defaultrate) As maxrate
      From employeeclass
```

Figura 9.18 prezintă rezultatele acestei interogări.

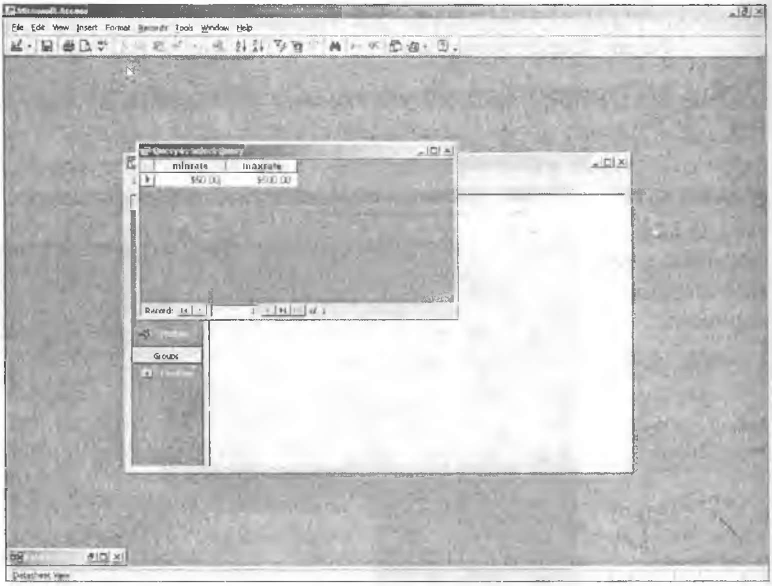


FIGURA 9.18
Funcțiile `Min()` și `Max()` returnează valorile minimă, respectiv maximă ale expresiei din argument.

A doua filtrare a rezultatelor: utilizarea clauzei `Having`

Adesea, va trebui să filtrați rezultatele pe baza datelor agregate. De exemplu, poate doriți să vedeți numai înregistrările în care un angajat a generat mai mult de 1000 $. Codul următor utilizează clauza Having pentru a obține rezultatele dorite:

```
SELECT employee.firstname+' '+employee.lastname AS employee,
       Sum(timeentrydetail.hours*timeentrydetail.rate) AS total
      FROM Employee
        INNER JOIN TimeEntryDetail ON Employee.EmployeeID =
TimeEntryDetail.EmployeeID
        GROUP BY employee.firstname+' '+employee.lastname
       Having Sum(timeentrydetail.hours*timeentrydetail.rate)>1000
       Order By Sum(timeentrydetail.hours*timeentrydetail.rate) Desc
```

Figura 9.19 prezintă rezultatele acestei interogări.

Clauza `Having` funcționează pe baza datelor agregate. Dacă ați fi încercat să utilizați clauza `Where`, SQL ar fi generat o eroare. Pe de altă parte, folosind clauza `Where`, ca de exemplu, `Where timeentrydetail.hours * timeentrydetail.rate > 1000`, criteriul ar fi fost aplicat numai pentru înregistrările din tabelul `TimeEntryDetail`. Orice înregistrare care nu depășește pragul de 1000 $ ar fi fost ignorată. În acest caz, doriți mai întâi să agregați toate înregistrările privind timpul necesar cazurilor, apoi să aplicați criteriul de filtrare.

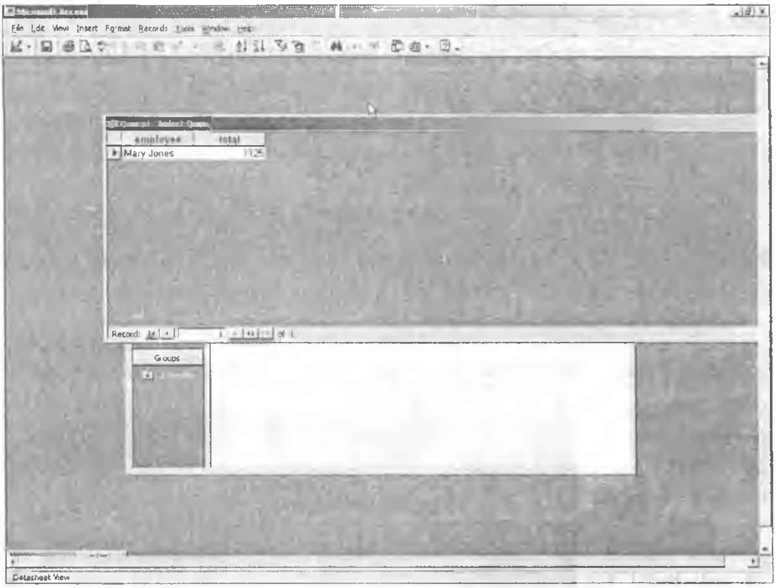


FIGURA 9.19
Clauza `Having` vă permite să filtrați setul de rezultate pe baza datelor agregate.

> **Indiciu**
> Dacă doriți să fie aplicat criteriul fiecărei surse de înregistrări, utilizați clauza `Where`. Dacă doriți să aplicați criteriul de filtrare rezultatelor obținute prin agregare, utilizați clauza `Having`.

A treia filtrare a rezultatelor: crearea declarațiilor SQL parametrizate

În exemplul precedent, declarația SQL a avut ca scop să returneze date sumarizate numai pentru cazul de divorț Smith. Adesea, veți dori să lăsați utilizatorul să specifice cazul. Problema aceasta apare în două arii importante din dezvoltarea aplicațiilor pentru baze de date.

Evident, este o problemă pentru baza de date. Este și o problemă pentru dezvoltarea interfeței cu utilizatorul. Problemele privind interfața cu utilizatorul sunt discutate în partea a cincea a cărții. Problemele privind modul în care baza de date se confruntă cu aceste declarații „dinamice în SQL" vor fi discutate acum.

Declarațiile SQL care solicită utilizatorului introducerea de date se numesc interogări parametrizate. Definiția declarației SQL conține un placeholder în locul unei valori statice. Pentru a ilustra acest lucru, să considerăm codul următor:

```
SELECT Case.Title,
    Sum(timeentrydetail.hours*timeentrydetail.rate) AS total
  FROM Case
    INNER JOIN TimeEntryDetail ON Case.CaseID=TimeEntryDetail.CaseID
    WHERE case.title=[introduceti numele unui caz]
    GROUP BY Case.Title
    ORDER BY Sum(timeentrydetail.hours*timeentrydetail.rate) Desc
```

Figura 9.20 arată ce se întâmplă la executarea acestei interogări. Va apărea o casetă de dialog care vă permite să specificați numele cazului.

FIGURA 9.20

Interogările parametrizate permit utilizatorului să specifice criteriul înainte de execuție.

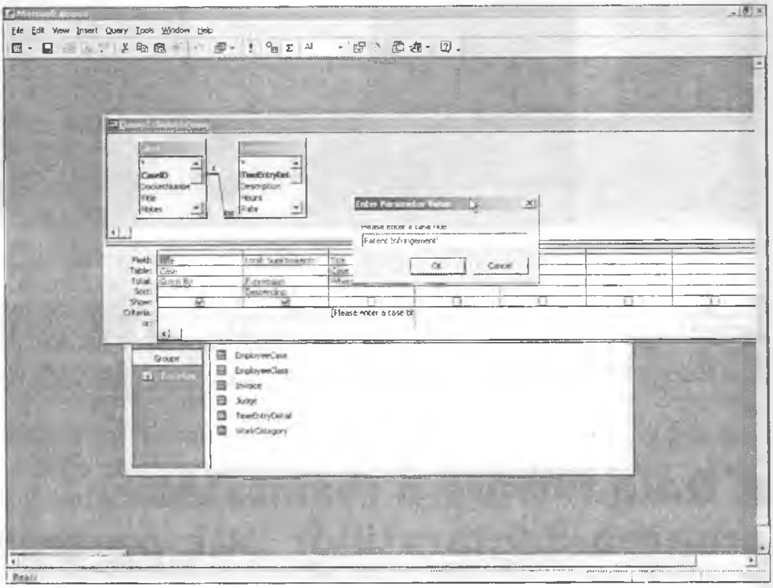


Figura 9.21 prezintă rezultatele interogării după ce utilizatorul a introdus o valoare.

În capitolul anterior v-a fost prezentat caracterul wildcard *, utilizat în Access. Interogările parametrizate pot folosi și ele acest caracter. Singura diferență este că, în locul operatorului =, trebuie să folosiți operatorul `Like`. Figura 9.22 prezintă o interogare parametrizată care implementează o căutare wildcard, iar figura 9.23 prezintă rezultatele interogării.

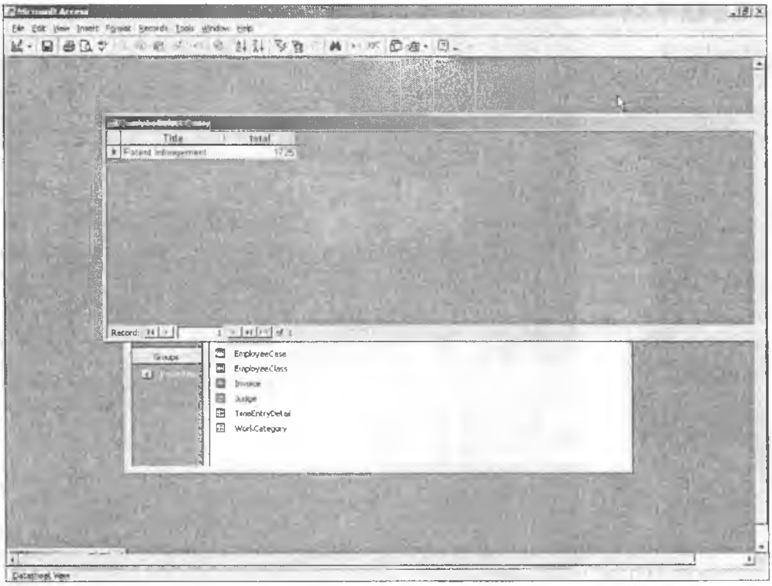


FIGURA 9.21 Acest set de rezultate a fost generat de o interogare parametrizată.

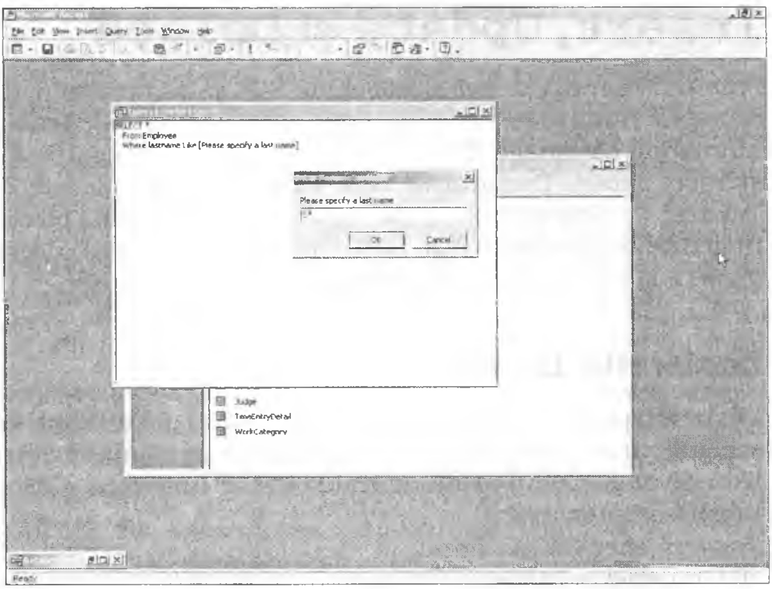


FIGURA 9.22 Operatorul Like și caracterul wildcard pot fi utilizate în interogările parametrizate.

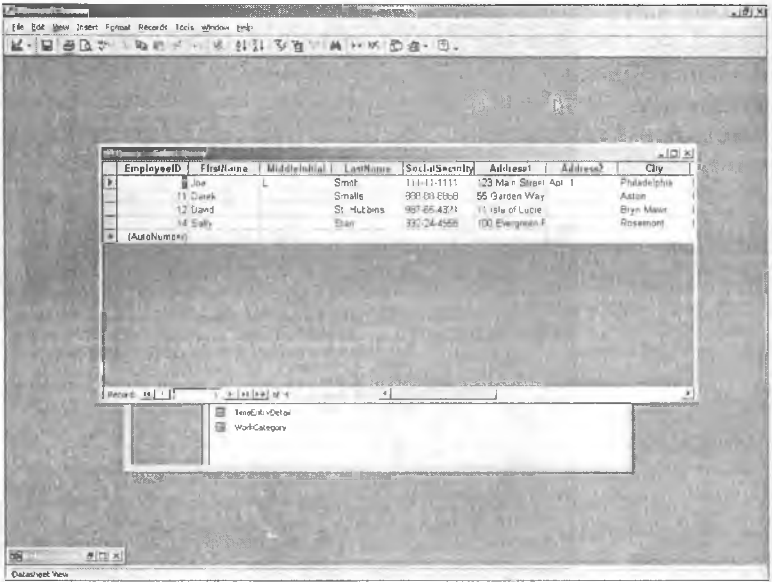


FIGURA 9.23 Rezultatele interogării parametrizate în care a fost utilizat caracterul wildcard pentru a efectua o căutare a numelor de tipul S*.

Modificarea datelor folosind declarațiile `Insert`, `Update` și `Delete`

Prezentarea introductivă a limbajului SQL este aproape încheiată. Singurele elemente rămase nediscutate sunt acele porțiuni ale limbajului Data Manipulation Language (DML) care vă permit să adăugați, să modificați și să ștergeți datele. Declarațiile SQL care facilitează aceste operațiuni sunt `Insert`, `Update` și `Delete`. Această secțiune prezintă o scurtă introducere în aceste componente SQL DML. Așa cum veți vedea, declarațiile `Insert`, `Update` și `Delete` sunt foarte simple și ușor de înțeles.

Declarația `Insert`

Declarația `Insert` este alcătuită din trei părți. Prima dintre ele desemnează tabelul cu care lucrați. A doua este o listă de câmpuri. A treia parte este o listă de date. Codul următor arată cum ar funcționa o declarație Insert în tabelul Department:

```
Insert Into Department
      (Description)
      Values ("Criminal Law")
```

Figura 9.24 prezintă modul în care este executată această interogare în Access.

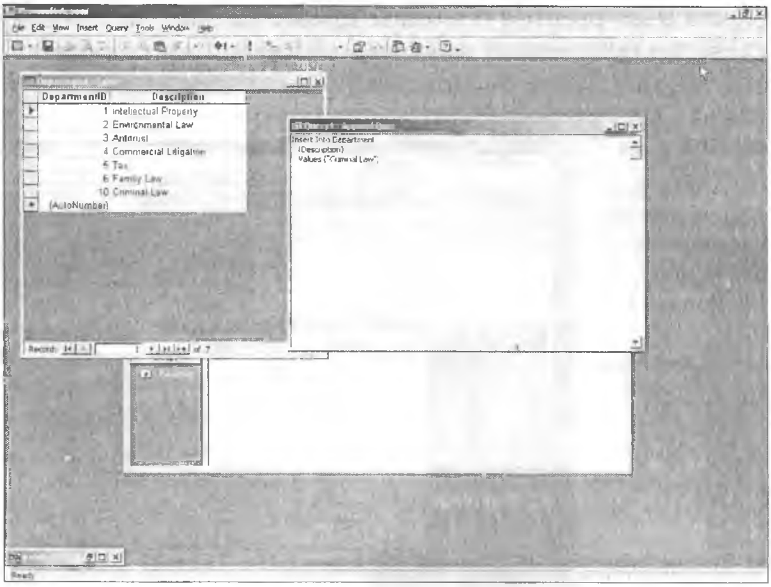


FIGURA 9.24
Prin executarea declarației SQL `Insert`, a fost introdus departamentul Criminal Law.

Notă

Unele baze de date acceptă următoarea declarație:

```
Insert Into Department
      Values ("Criminal Law")
```

Cu alte cuvinte, unele baze de date vă permit să omiteți o listă explicită de câmpuri. Access, totuși, necesită o listă de câmpuri. Omiterea acestei liste nu este o idee bună, deoarece faceți presupunerea că structura de bază a tabelului nu s-a modificat. Dacă structura nu s-a modificat, este foarte posibil să obțineți o eroare prin codul SQL. Explicitând lista de câmpuri, codul dumneavoastră este mai lizibil și mai ușor de întreținut. Iar dacă tabelul utilizează valorile implicite, trebuie numai să precizați câmpurile în care trebuie completată o valoare.

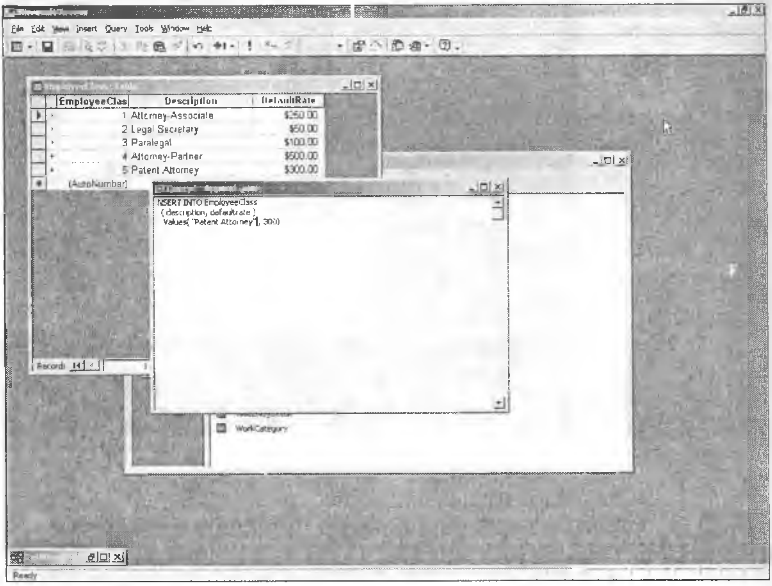
Indiciu

După ce introduceți în tabel o nouă valoare, cum puteți determina valoarea cheii ei primare? Nu uitați, sistemul generează valoarea cheii. Întrucât sunteți sigur că aceasta este valoarea maximă din tabel, funcția `Max()` va rezolva problema:

```
Select Max(departmentid) As lastkey
      From department
```

Figura 9.25 prezintă o situație în care în tabel sunt introduse mai multe valori. Se pune numai problema afișării tuturor câmpurilor cu care lucrați și a valorilor corespondente acestora în câmpuri.

FIGURA 9.25
Declarația SQL `Insert` permite introducerea mai multor valori în noul rând `EmployeeClass`.



Declarația `Update`

Declarația Update conține două părți importante. Prima parte desemnează tabelul cu care lucrați. A doua este o listă a câmpurilor și a noilor valori care trebuie introduse în ele. Codul următor mărește cu 10 % valoarea tarifului implicit din tabelul `EmployeeClass`:

```
Update EmployeeClass
      Set DefaultRate = DefaultRate * 1.10
```

Figura 9.26 prezintă modul în care este executată această interogare în Access.

Declarația `Update` poate fi completată cu o clauză `Where`. De exemplu, dacă doriți să măriți cu 10 % numai tarifele care depășesc 500 $, trebuie să scrieți un cod ca acesta:

```
Update Employeeclass
      Set DefaultRate = DefaultRate * 1.10
      Where DefaultRate>500
```

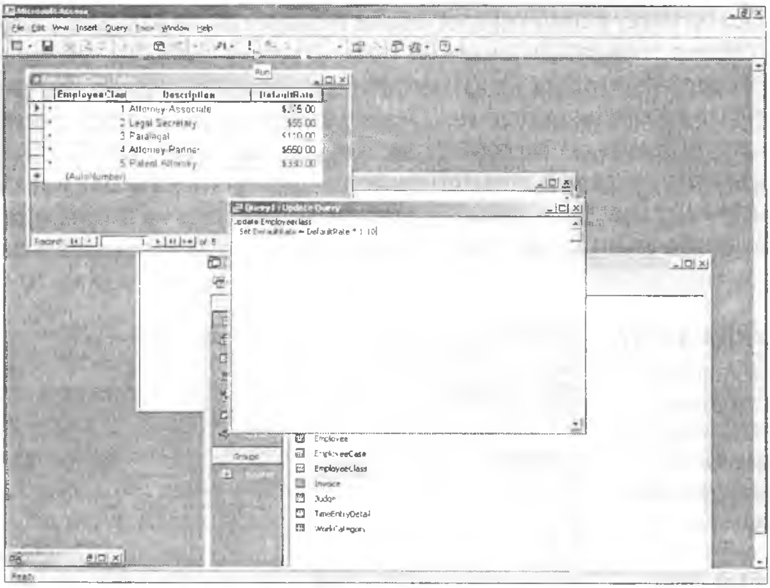


FIGURA 9.26
Valorile tarifului implicit au fost modificate prin declarația SQL `Update`.

Dacă trebuie actualizate valorile din mai multe câmpuri, trebuie numai să separați prin virgulă numele acestor câmpuri și valorile lor din lista scrisă în declarație. Prototipul următor poate ilustra modul în care poate fi realizată această sarcină:

```
Update <<numeletabelului>>
      Set <<campulx>> = <<noua valoare>>,
          <<campuly>> = <<noua valoare>>,
          <<campulz>> = <<noua valoare>>
      where <<camp>> = <<valoare>>
```

Declarația `Delete`

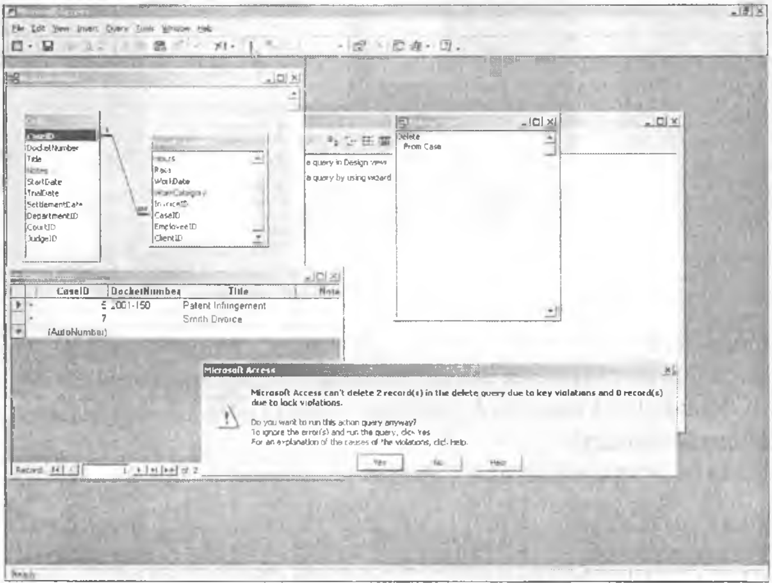
Declarația `Delete` are două părți importante. Prima, ca și în cazul declarațiilor `Insert` și `Update`, este tabelul cu care lucrați. A doua este clauza `Where` pentru a stabili rândurile care trebuie șterse. Codul următor va încerca să șteargă toate înregistrările din tabelul `Case`:

```
Delete
      From Case
```

Întrucât nu există clauza `Where` care să stabilească datele care trebuie șterse, SQL presupune că doriți să ștergeți toate datele. Acest lucru este valabil și pentru declarația `Update`. În cazul actualizărilor, dacă nu precizați datele care trebuie actualizate, SQL presupune că doriți să le actualizați pe toate.

Din fericire, există un mecanism încorporat care vă ajută să evitați compromiterea integrității bazei de date. Dacă vă amintiți, în capitolul 7 v-a fost prezentat conceptul integrității referențiale. Anexa B, conține o listă a relațiilor din baza de date. Despre tabelul `Case` putem spune că are o relație unu la mulți cu tabelul `TimeEntryDetail`. Întrucât în baza de date sunt utilizate ștergerile restricționate, iar pentru fiecare caz există înregistrări în tabelul `TimeEntryDetail`, ștergerea unui caz va fi imposibilă. Figura 9.27 arată ce se întâmplă dacă încercați să ștergeți înregistrări din tabelul Case.

FIGURA 9.27
Regulile de integritate referențială vă împiedică să ștergeți date, adică să compromiteți integritatea referențială a bazei de date.



Deși caseta de dialog din figura 9.27 vă întreabă dacă doriți să continuați, înregistrările nu vor fi șterse. Regulile de integritate referențială configurate de dumneavoastră pentru baza de date sunt imuabile.

Codul următor oferă un ultim exemplu, care încorporează clauza `Where` în declarația `Delete`:

```
Delete
      From EmployeeClass
      Where EmployeeClassID = 5
```

Dacă în clasa de angajați pe care doriți să o ștergeți nu există angajați, operația de ștergere va reuși, deoarece integritatea referențială nu este încălcată în acest caz.

Rezumat

În acest capitol, am extins discuția despre conceptele SQL învățate în capitolul 8. Capitolul s-a concentrat asupra unor utilizări avansate ale limbajului SQL. Va trebui adesea să agregați datele folosind funcțiile `Sum()`, `Count()` și `Avg()`. În funcție de gradul necesar de granularitate a datelor, va trebui să încorporați și clauza `Group By`. În plus, funcțiile `Min()` și `Max()` oferă o cale simplă de a extrage valorile minimă și maximă ale unui câmp dat dintr-un tabel.

După agregarea datelor, este foarte posibil să doriți să filtrați setul de rezultate pe baza rezultatelor calculate. Așa cum ați învățat, clauza `Where` nu vă va ajuta în această situație, iar în locul ei va trebui să utilizați clauza `Having`. Clauza `Where` acționează asupra datelor sursă în timpul execuției interogării, iar clauza `Having` acționează asupra setului de rezultate obținute după terminarea execuției interogării. În final, ați învățat interogările parametrizate și alte comenzi din sublimbajul DML din SQL: `Insert`, `Update` și `Delete`.

Acum, când ați înțeles elementele esențiale ale limbajului SQL, puteți utiliza instrumentele grafice pentru interogări din Access. Scopul acestui capitol și al celui precedent este să vă ajute să înțelegeți SQL ca limbaj, independent de implementarea sa într-un caz particular. După ce ați înțeles limbajul, este mult mai ușoară adoptarea instrumentelor grafice care vă ajută să evitați complexitatea codurilor SQL. De asemenea, pe măsură ce faceți progrese pe calea dezvoltării aplicațiilor pentru baze de date, va trebui să recurgeți adesea la codul SQL inclus în aplicații. Mai simplu spus, dacă aveți de gând să deveniți dezvoltator de aplicații pentru baze de date sau administrator al bazelor de date, trebuie să cunoașteți și să utilizați limbajul SQL!

10



INCLUDEREA INTEROGĂRILOR SQL ÎN BAZA DE DATE

În acest capitol:

- Prezentare generală a aplicației Query Designer din Access
- Crearea interogărilor avansate
- Rezumat

În ultimele două capitole ați învățat despre funcțiile de bază și funcțiile avansate ale limbajului SQL (Structured Query Language). În acest capitol, veți învăța cum puteți să stocați interogări în baza dumneavoastră de date. Stocarea interogărilor în baza de date are multe avantaje. Unul dintre ele este faptul că puteți realiza ceea ce se numește independența program / date (*program/data independence*). Aceasta înseamnă că modificările în codul dumneavoastră de program nu presupun neapărat modificări în baza de date și viceversa. O bună proiectare implică separarea codului de program și a metodelor specifice de acces și manipulare a datelor.

Scopul acestei cărți cuprinde aspectul legat de date al problemei, dar aceasta nu înseamnă că nu i se acordă atenție și limbajului de programare. După ce înțelegeți limbajul SQL, vi se pare că este la fel cu orice alt program. Ultimele două capitole v-au prezentat utilizarea propriu-zisă a acestui limbaj, independent de o implementare particulară într-o bază de date. Pe măsură ce vă implicați mai mult în dezvoltarea aplicațiilor pentru baze de date, veți vedea că este imposibil să stocați codul SQL direct în logica aplicației dumneavoastră. În acest capitol veți afla cum puteți stoca direct într-o bază de date codul SQL. În timp ce includerea codului SQL în logica aplicației violează independența date / program, stocarea codului SQL în baza de date o încurajează.

Stocarea interogărilor într-o bază de date depinde de zonele specifice implementării unei baze de date. În acest capitol, veți învăța cum vă permite Access să stocați interogările. Alte programe pentru baze de date, ca SQL Server și Oracle au metodologii proprii.

O prezentare generală a aplicației Query Designer din Access

Credeți sau nu, dar ați lucrat deja cu aplicația Query Designer din Access! Chiar în capitolul 8, ați lucrat în vizualizarea SQL a acestei aplicații. Când selectați Queries din lista de obiecte a bazei de date, apar două opțiuni în panoul din dreapta al ferestrei principale a bazei de date. Când optați pentru crearea unei interogări în vizualizarea Design, în plus față de Query Designer, apare o casetă de dialog care vă invită să selectați tabelele care vor fi introduse în interogare. Figura 10.1 prezintă caseta de dialog Show Table.

Indiciu

Caseta cu listă din caseta de dialog ShowTable funcționează ca orice altă casetă cu listă din Windows. Dacă țineți apăsată tasta Control (Ctrl) în timp ce selectați elementele, vor fi selectate toate elementele pe care ați executat Ctrl+clic. Dacă selectați un element și țineți apăsată tasta Shift când selectați alt element, vor fi selectate toate elementele cuprinse între cele două pe care ați executat clic.

După selectarea unui tabel, acesta apare în panoul de sus al aplicației Query Designer. Pentru a exemplifica, să folosim tabelul Client. După selectarea acestui tabel, el apare în panoul de sus al aplicației. Tabelul Client este prezentat în figura 10.2.

FIGURA 10.1

Aplicația Query Designer din Access.

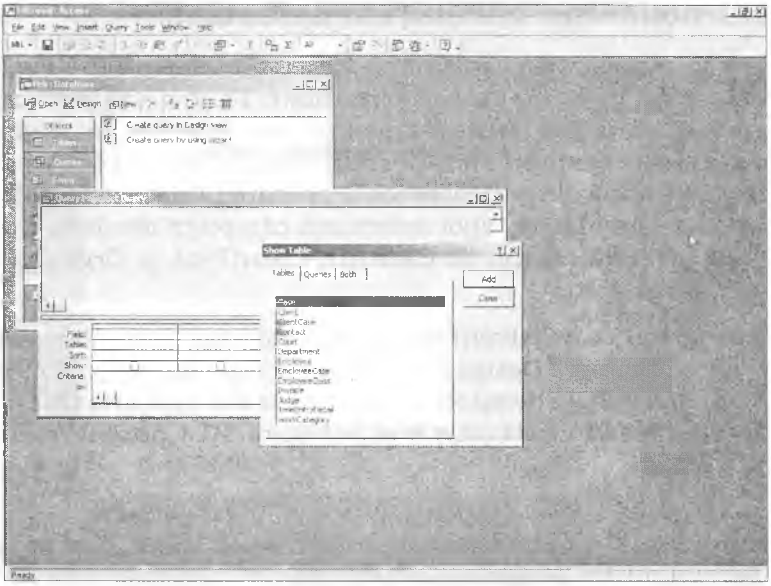
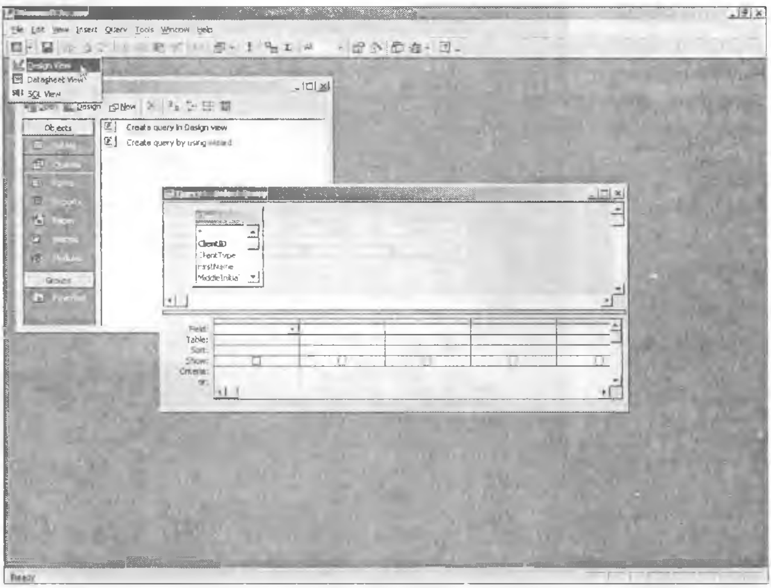


FIGURA 10.2

Aplicația Query Designer din Access în vizualizarea Design.

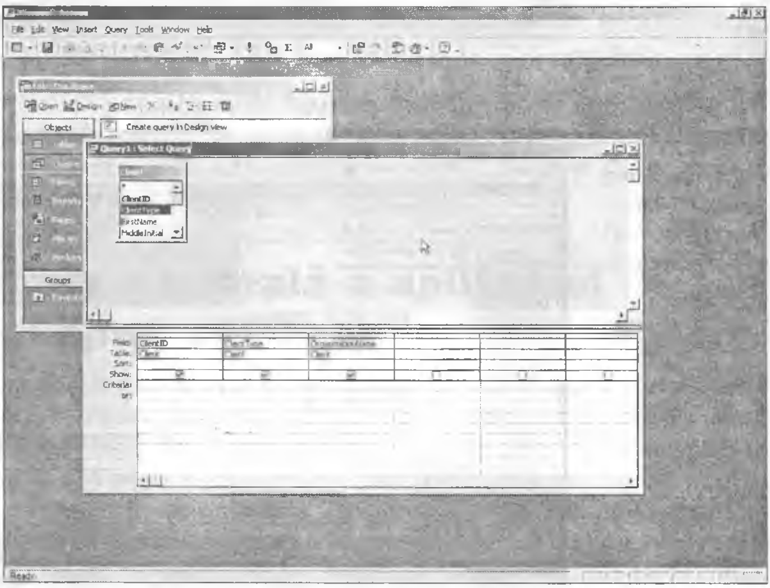


Introducerea câmpurilor selectate

După introducerea tabelului în Query Designer, puteți selecta câmpurile care vor fi incluse în setul de rezultate. Pentru a selecta un câmp, trebuie doar să executați dublu clic pe el în lista câmpurilor. Altfel, îl puteți trage din lista de câmpuri ale tabelului, aflată în panoul de sus și îl puteți plasa într-o coloană din panoul de jos al aplicației. O a treia metodă este selectarea unei coloane libere, apoi selectarea câmpului din lista derulantă. Figura 10.3 prezintă câmpurile ClientID, ClientType și OrganizationName după selectare.

Pe măsură ce construiți interogarea folosind aplicația grafică, puteți trece de la vizualizarea Design la vizualizarea SQL. Ar trebui să fiți deja familiarizat cu vederea SQL, deoarece ea face parte din aplicația Query Designer și ați lucrat cu ea în capitolele 8 și 9. Figura 10.4 prezintă vizualizarea SQL a interogării.

FIGURA 10.3
Câmpurile ClientID, ClientType și Organization-Name au fost selectate pentru această interogare.



După ce ați selectat câmpurile, puteți rula interogarea și veți vedea setul de rezultate. Rezultatele curente sunt prezentate în figura 10.5.

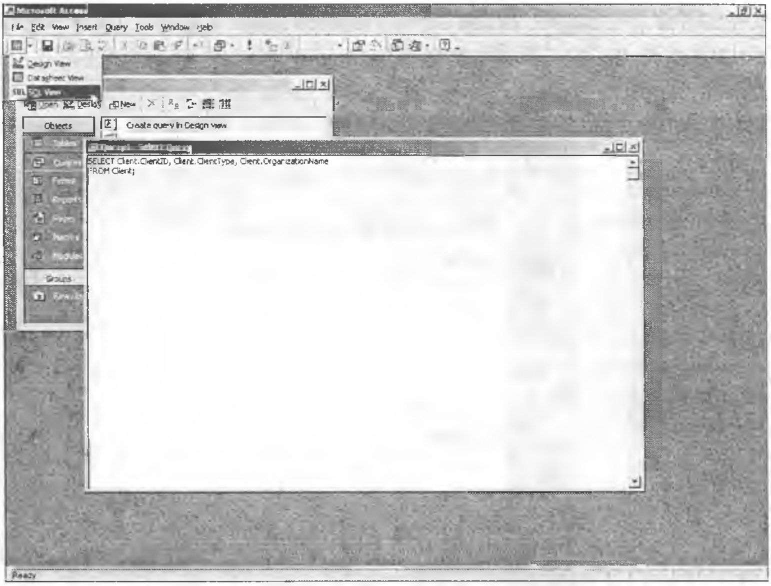


FIGURA 10.4
Trecerea la vizualizarea SQL vă permite să vedeți codul SQL.

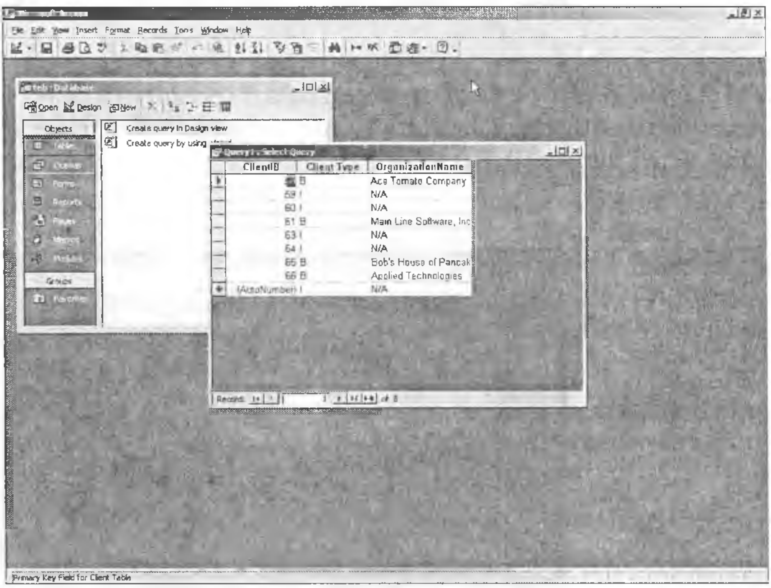


FIGURA 10.5
Puteți rula interactiv interogarea în aplicația Query Designer pentru a vedea conținutul setului de rezultate.

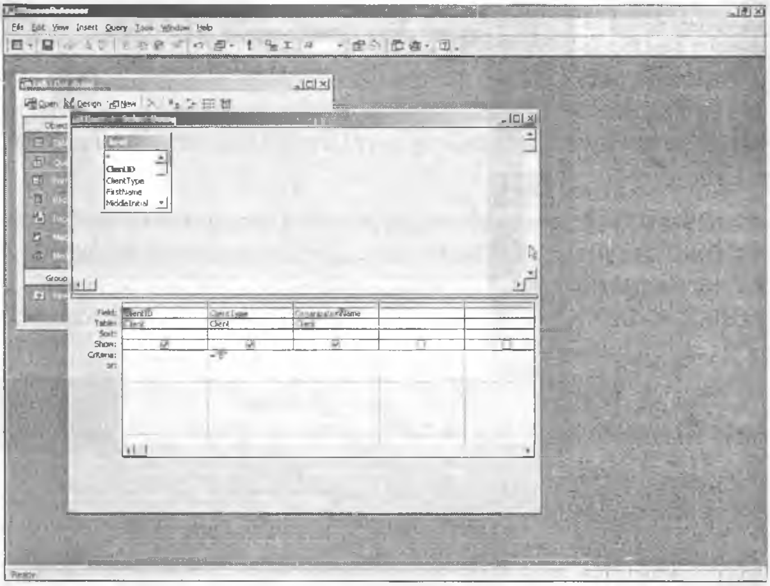
Introducerea criteriului de selecție

După ce ați selectat câmpurile dorite, puteți filtra setul de rezultate folosind un criteriu de selecție. Dacă priviți figura 10.5, este clar că înregistrările

clienților persoane fizice sunt nerelevante. Figura 10.6 prezintă criteriul de selecție în aplicația de proiectare grafică.

FIGURA 10.6

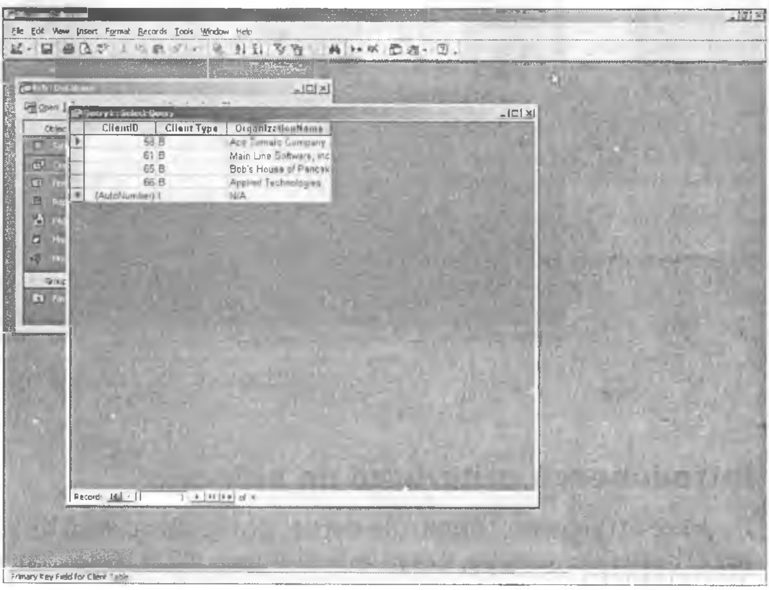
Criteriul de selecție a fost introdus pentru ca interogarea să returneze numai înregistrările care îndeplinesc condiția ClientType = "B".



Ca și în exemplul precedent, puteți rula interogarea în mod interactiv pentru a vedea rezultatele. Rezultatele care respectă criteriul de selecție sunt prezentate în figura 10.7.

FIGURA 10.7

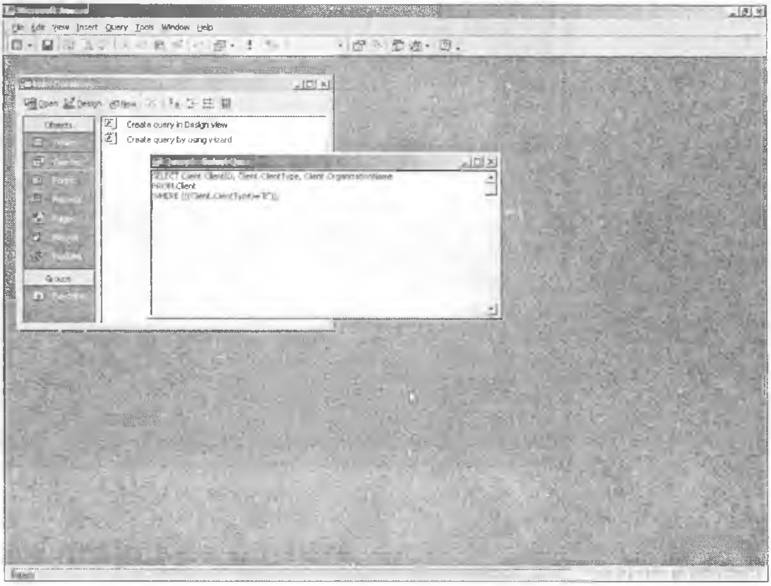
Setul de rezultate reflectă introducerea criteriului de selecție.



În final, trecând în vizualizarea SQL, puteți vedea codul SQL. Codul corespunzător acestei interogări este prezentat în figura 10.8.

FIGURA 10.8

Introducerea criteriului de selecție în aplicația grafică de proiectare duce la introducerea clauzei `Where` în codul SQL.



Suprimarea coloanelor

Singurul scop al câmpului ClientType este filtrarea setului de rezultate. Afișarea acestei coloane în setul de rezultate nu este necesară. Între porțiunile Sort și Criteria ale panoului de jos al aplicației de proiectare, veți vedea o casetă de validare prin care puteți stabili dacă o coloană este returnată în setul de rezultate. Figura 10.9 vă arată cum puteți suprima coloana ClientType din setul de rezultate, iar figura 10.9 prezintă setul de rezultate modificat.

Figura 10.11 prezintă codul SQL corespunzător acestei interogări. După cum vedeți, suprimarea unei coloane înseamnă doar ștergerea câmpului din lista de selecție.

FIGURA 10.9

Puteți stabili dacă o coloană apare sau nu în setul de rezultate.

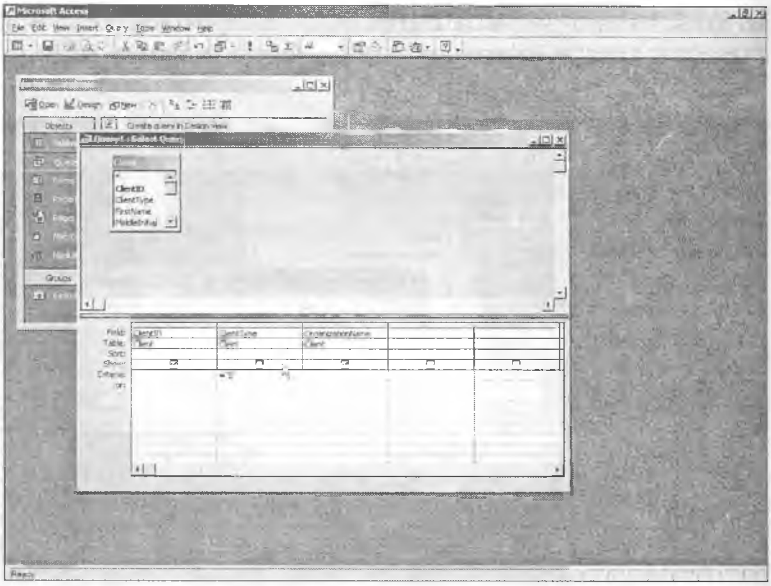
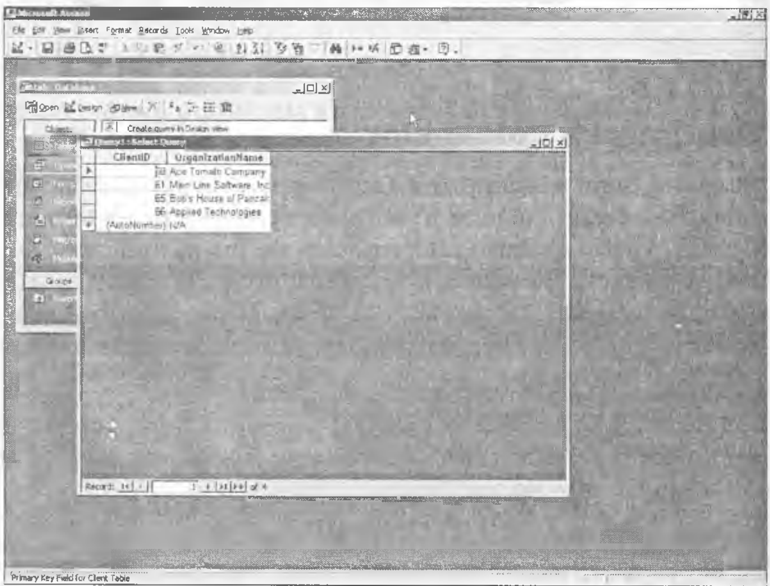


FIGURA 10.10

Setul de rezultate reflectă ștergerea câmpului ClientType.



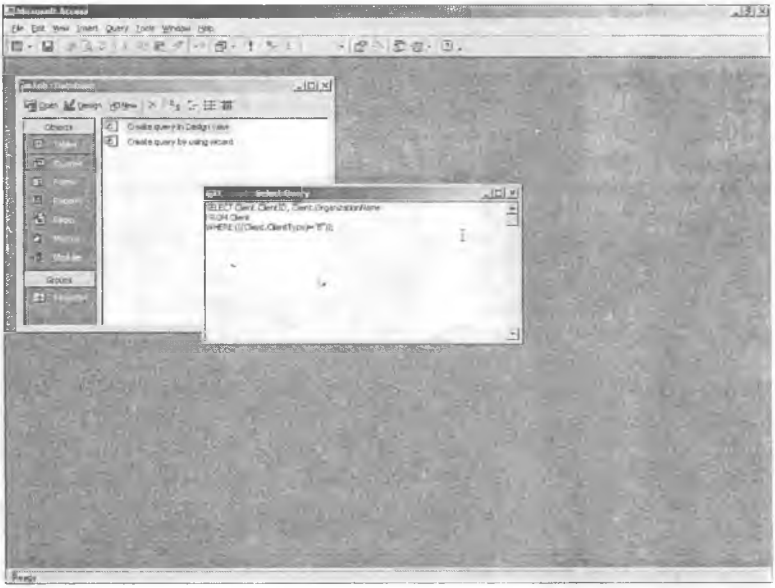


FIGURA 10.11
Suprimarea unei coloane înseamnă doar ștergerea câmpului din lista de selecție.

Introducerea ordinii de sortare

Singura sarcină importantă pe care o mai aveți este să furnizați ordinea de sortare. Figura 10.12 arată cât de ușor este să aplicați o ordine de sortare după câmpul OrganizationName. Figura 10.13 prezintă setul de rezultate, iar figura 10.14 prezintă codul SQL corespunzător.

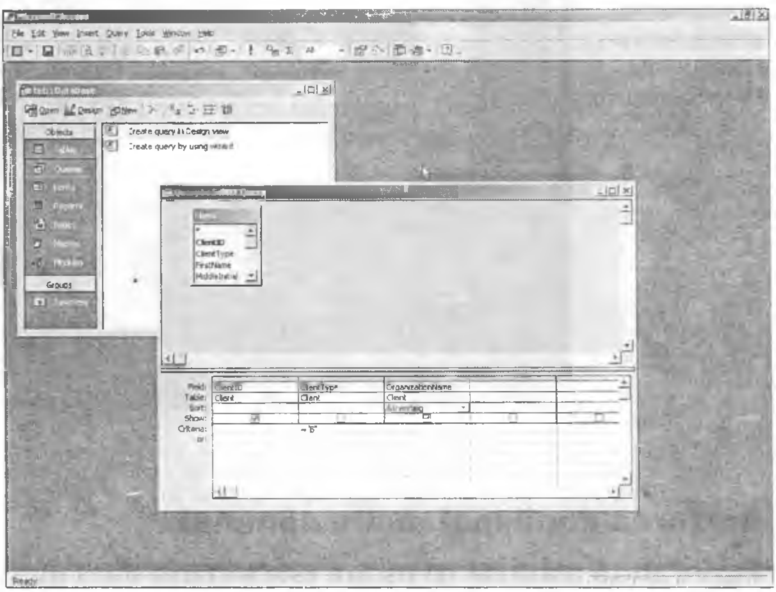


FIGURA 10.12
Puteți specifica, indiferent de câmpul ales pentru sortare, dacă sortarea este ascendentă sau descendentă.

FIGURA 10.13 Setul de rezultate prezintă sortarea ascendentă după câmpul Organization-Name.

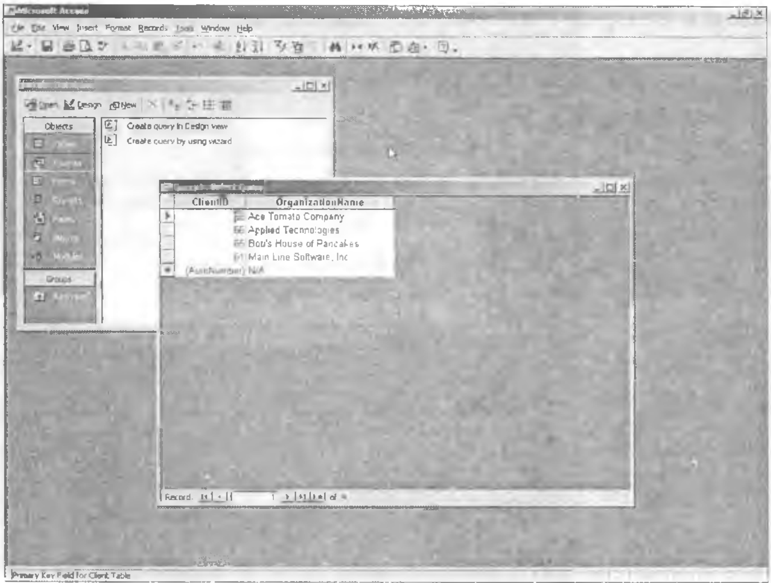
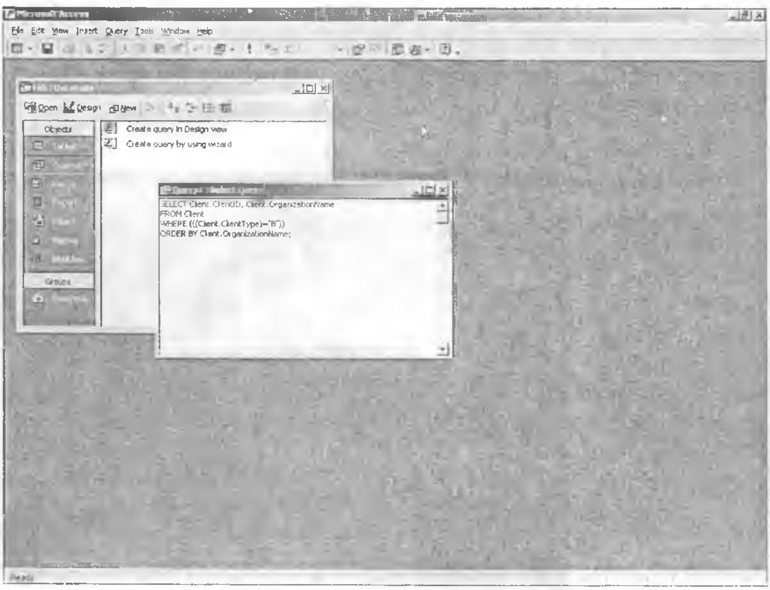


FIGURA 10.14 Specificarea ordinii de sortare în aplicația de proiectare grafică duce la introducerea clauzei Order By în codul SQL.

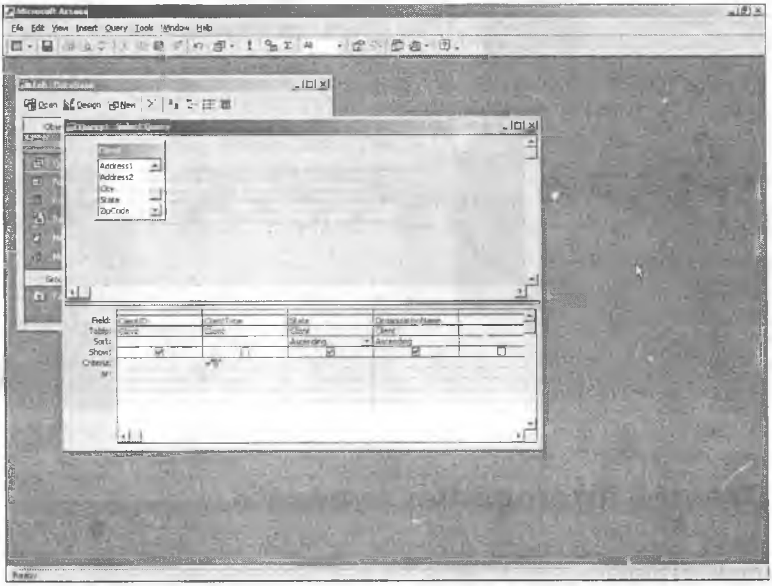


Sortarea după mai multe câmpuri

Figura 10.15 arată cât de ușoară este sortarea după mai multe câmpuri. Aceasta înseamnă numai specificarea ordinii ascendente sau descendente

pentru fiecare câmp sortat. Precedența sortării funcționează de la stânga la dreapta. În această situație, setul de rezultate va fi sortat mai întâi după stat, apoi după numele organizației. Pentru a suporta această sortare suplimentară, a fost introdus în interogare câmpul State.

FIGURA 10.15
Setul de rezultate este sortat mai întâi după stat, apoi după numele organizației.



Salvarea interogării

Salvarea interogării este ușoară. Trebuie doar să executați clic pe butonul Save de pe bara de instrumente. Figura 10.16 prezintă un mesaj care vă permite să specificați un nume pentru interogare. După salvarea interogării, dacă executați dublu clic pe ea în fereastra principală a bazei de date, aceasta va fi rulată.

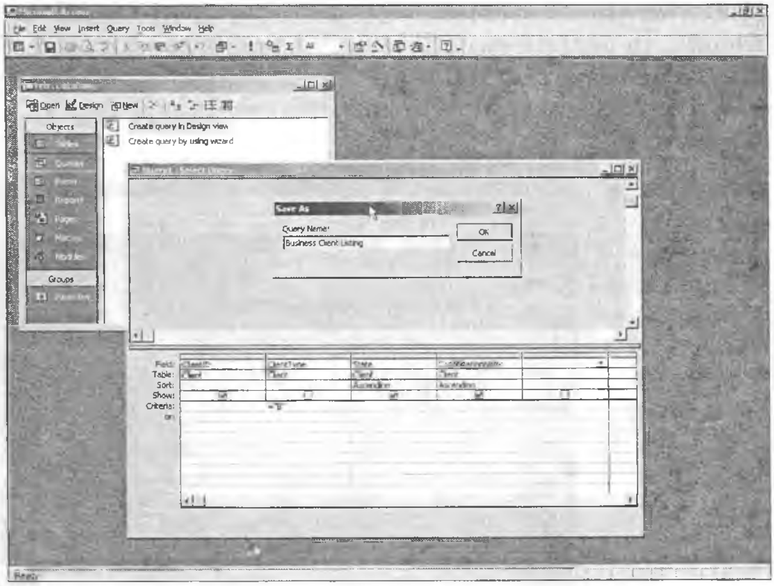


FIGURA 10.16
Caseta de dialog Save As vă permite să specificați numele interogării.

Crearea interogărilor avansate

Până acum ați văzut funcțiile de bază ale aplicație Query Designer din Access. Crearea interogărilor avansate folosind această aplicație este aproape la fel de ușoară ca și crearea interogărilor obișnuite. Să considerăm o listă simplă a angajaților, care conține câmpurile EmployeeID, EmployeeClass, FirstName și LastName. Așa cum știți deja, descrierea clasificării angajaților apare și în tabelul EmployeeClass. Aceasta înseamnă că interogarea necesită două tabele și o declarație de uniune. Întrucât relația este stocată în baza de date, nu trebuie să vă amintiți detaliile despre uniune. Când creați interogarea și introduceți tabelele Employee și EmployeeClass, uniunea va apărea automat. Figura 10.17 prezintă această interogare în aplicația de proiectare grafică, iar figura 10.18 prezintă rezultatele interogării.

FIGURA 10.17

Uniunile sunt create automat când în interogare sunt introduse tabele înrudite.

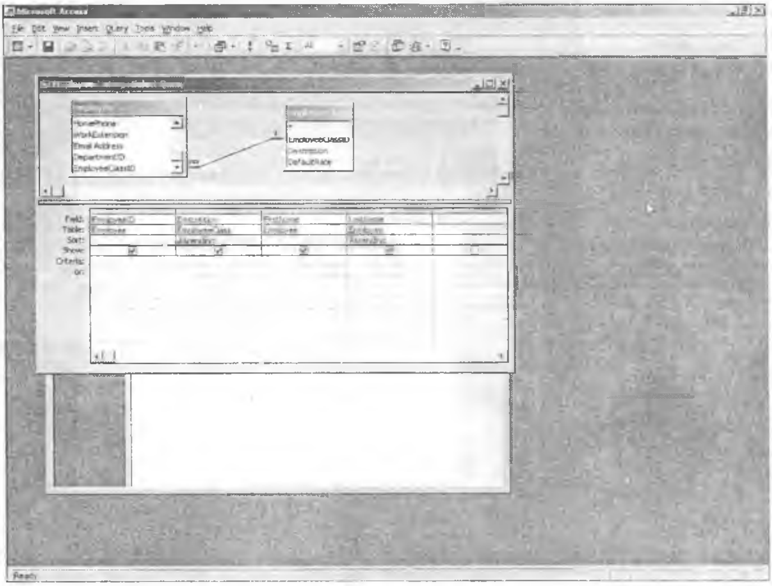


FIGURA 10.18

Setul de rezultate conține date din tabelele Employee și EmployeeClass.

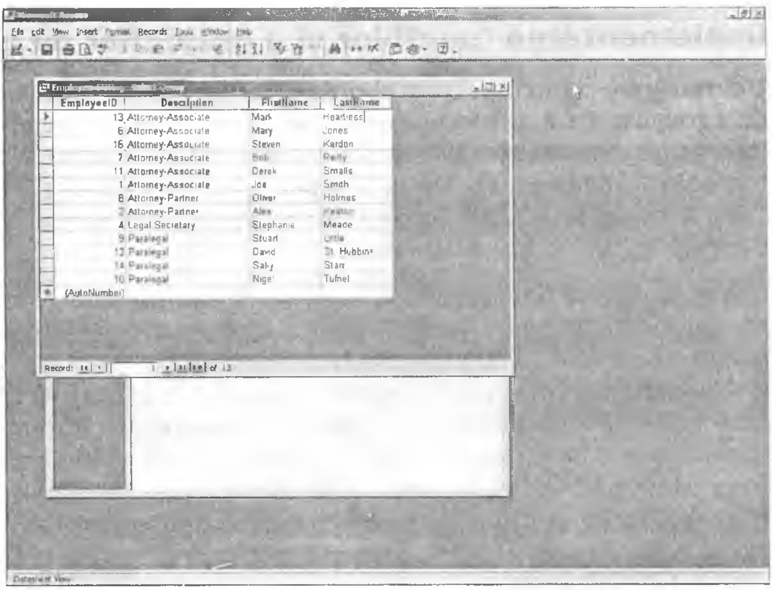


Figura 10.19 prezintă codul SQL pentru interogarea care returnează lista angajaților.

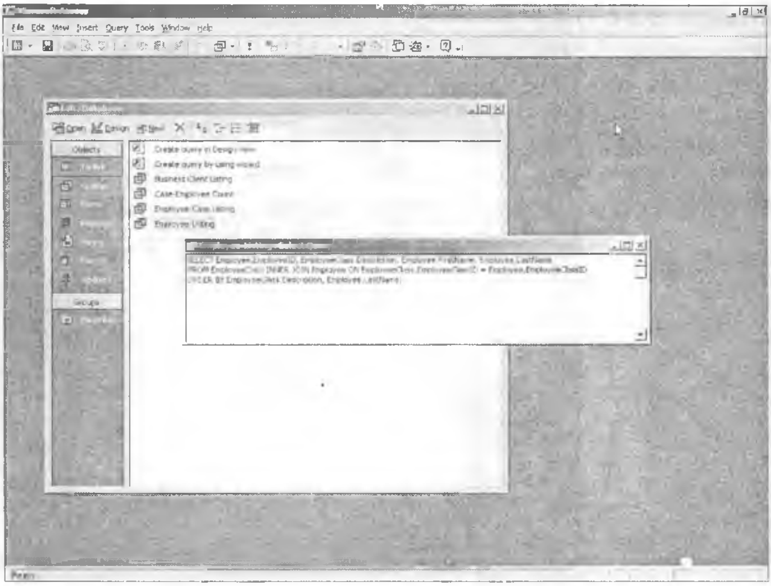


FIGURA 10.19 Vizualizarea SQL prezintă codul corespunzător interogării care returnează lista angajaților.

Implementarea funcțiilor și a grupărilor de agregare

Ultimul subiect pe care îl vom aborda este reprezentat de funcțiile și grupările de agregare. Pentru a exemplifica, să presupunem că trebuie să creați o interogare care realizează o listă a cazurilor și a numărului de angajați care lucrează la fiecare caz. De la început, știți că trebuie să folosiți tabelele Case și EmployeeCase. Întrucât numele angajaților nu prezintă interes, tabelul Employee nu este necesar.

Din aceste două tabele, veți avea nevoie de câmpul Title al tabelului Case și de funcția de agregare Count() pentru a număra angajații care lucrează la fiecare caz. Pentru a facilita numărarea, va trebui să folosiți o funcție „group by" pentru câmpul Title. Aplicația Query Designer nu are capacitatea implicită de a introduce grupări și funcții de agregare. Figura 10.20 prezintă două aspecte. Mai întâi, prezintă butonul Totals, de pe bara principală de instrumente. Când executați clic pe acest buton, în panoul din partea de jos a aplicației de proiectare a interogărilor apare o nouă secțiune, numită *Total*. Apoi, figura 10.20 vă prezintă interogarea în Query Designer.

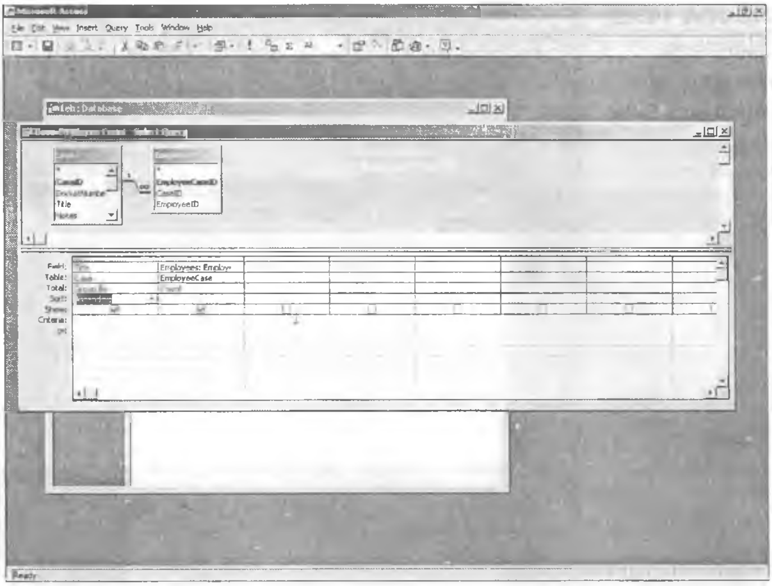


FIGURA 10.20

Această interogare implementează funcțiile de agregare Count() și Group By.

În secțiunea Total din Query Designer, puteți alege funcțiile de agregare specifice pe care doriți să le implementați. Figura 10.21 prezintă setul de rezultate, iar figura 10.22 prezintă codul SQL corespunzător.

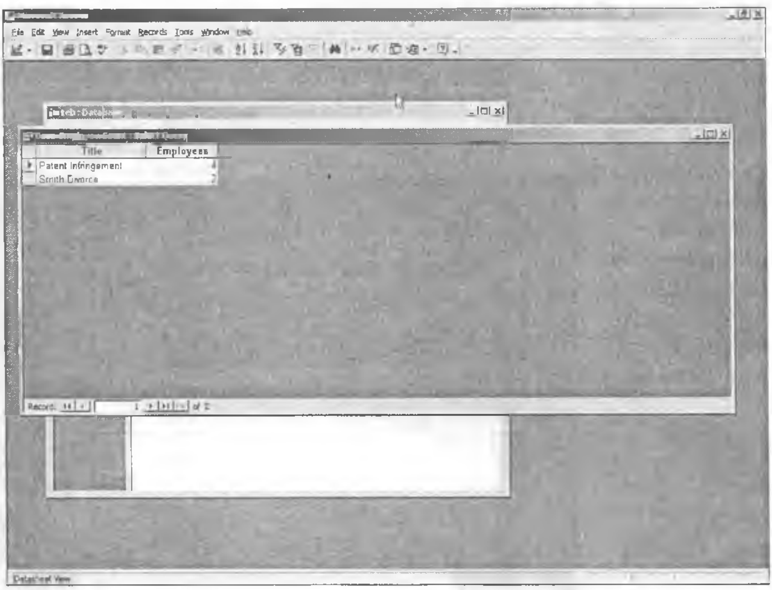


FIGURA 10.21

Setul de rezultate ale interogării Case Employee Count.

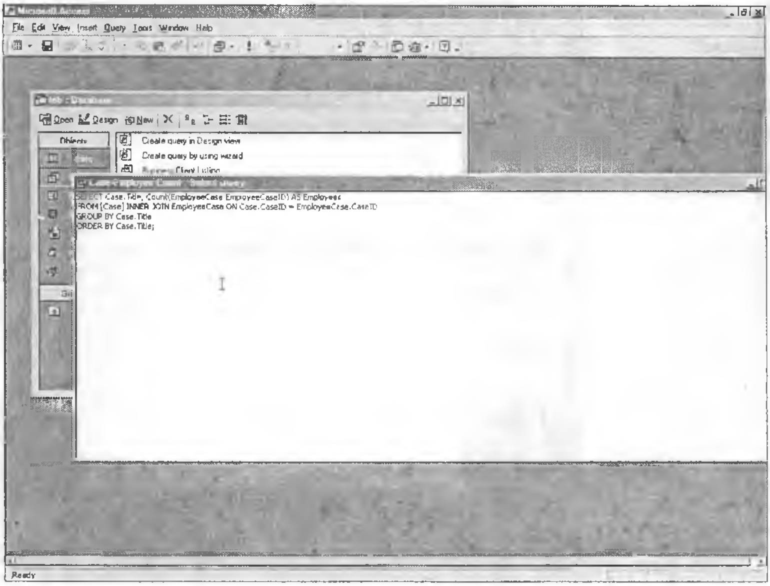


FIGURA 10.22
Codul SQL corespunzător interogării Case Employee Count.

Interogări parametrizate

Și interogările parametrizate sunt ușor de implementat. Pentru a ilustra acest lucru, să presupunem că doriți să creați o listă cu clienții care sunt firme. Diferența cheie este că, atunci când rulați interogarea, utilizatorul este invitat să introducă statul. Figura 10.23 prezintă această interogare în Query Designer.

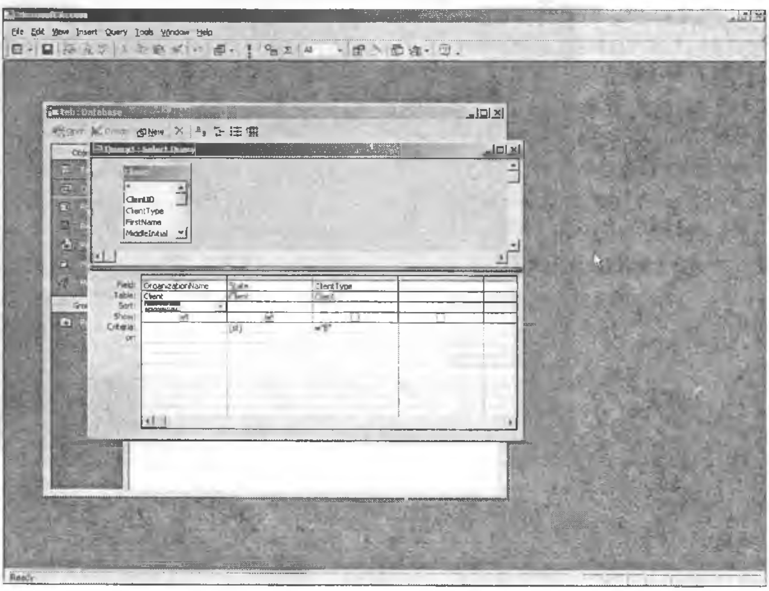


FIGURA 10.23
La rularea acestei interogări, utilizatorul va fi invitat să introducă o valoare pentru câmpul State.

Figura 10.24 arată cum utilizatorului i se solicită un parametru, iar figura 10.25 prezintă rezultatele interogării.

FIGURA 10.24

La rularea unei interogări parametrizate, utilizatorul este invitat să sprcifice o valoare pentru fiecare parametru.

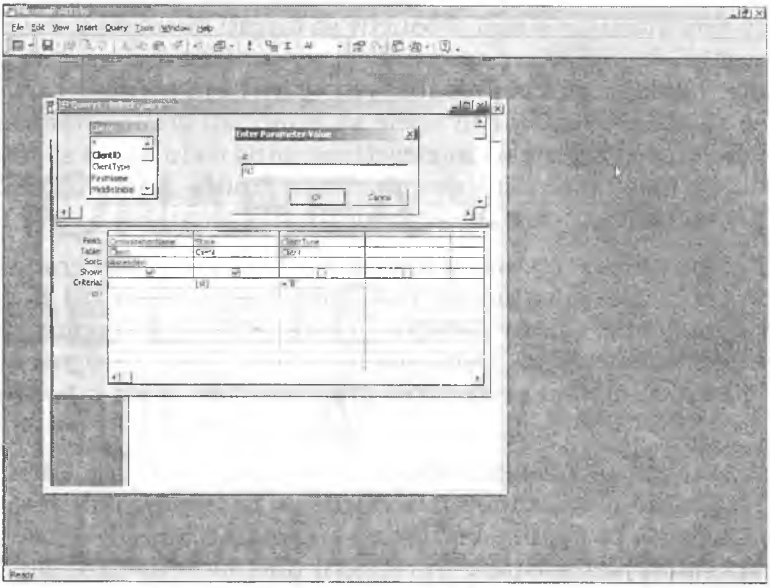
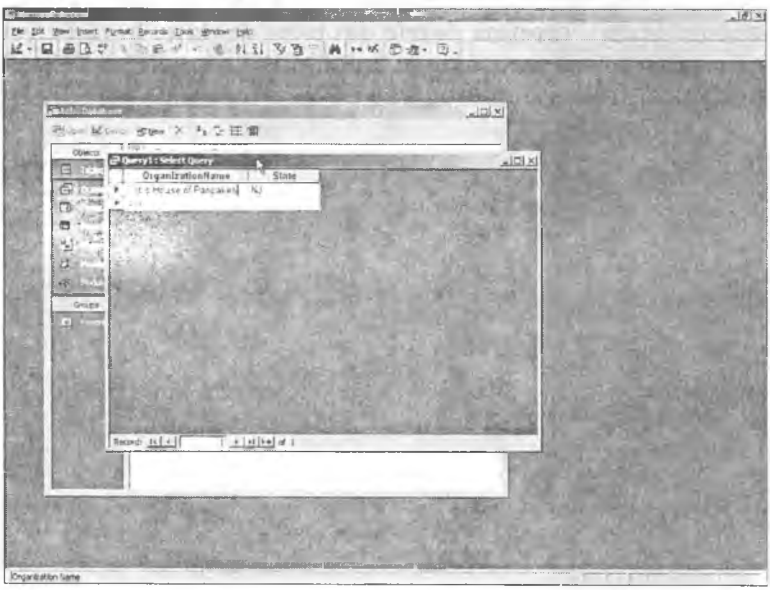


FIGURA 10.25

Setul de rezultate prezintă clienții care sunt firme din statul New Jersey.



Rezumat

Să sperăm că ați învățat că instrumentul grafic de proiectare a interogărilor este doar o extensie a limbajului SQL, pe care deja îl cunoașteți. Veți vedea că este mai ușor și mai productiv să lucrați direct cu codul decât cu instrumentul de proiectare sau poate vi se va părea mai ușor să proiectați grafic interogările. Cel mai probabil, veți vedea că în anumite cazuri este mai bine să utilizați codul, iar în altele să proiectați grafic interogările. Lucrul cel mai important este să înțelegeți mai întâi codul, apoi să înțelegeți funcționarea instrumentului de proiectare grafică a interogărilor. Acesta este motivul pentru care am structurat astfel capitolele 8, 9 și 10.

Indiferent dacă lucrați cu instrumentul de proiectare grafică sau în cod, tot în Query Designer lucrați. Este important să observați că cele două vederi ale aplicației Query Designer reprezintă două moduri de vizualizare a aceleiași interogări. Deoarece lucrați în Query Designer, indiferent dacă folosiți codul sau proiectarea grafică, puteți stoca interogarea în baza de date.

Acest capitol încheie abordarea primară a acestei cărți. Recapitulând, partea I a realizat o prezentare introductivă a bazelor de date iar partea a II-a v-a călăuzit în procesul proiectării și creării unei baze de date. Partea a III-a v-a prezentat mecanismele întreținerii unei baze de date, iar partea a IV-a a realizat o prezentare introductivă în limbajul SQL, pe care îl veți folosi pentru a extrage informații din baza de date. Partea următoare și ultima a cărții, partea a V-a, vă va arăta cum puteți construi interfețe cu utilizatorul simple și cum puteți întreține datele din baza de date. În plus, veți învăța și cum să construiți rapoarte simple. În timpul discuției despre rapoarte, veți vedea cum sunt utilizate interogările pe care tocmai le-ați construit.

PARTEA A V-A

FUNCȚIONAREA BAZEI DE DATE – CONSTRUIREA UNEI APLICAȚII SIMPLE PENTRU BAZE DE DATE ÎN ACCESS



11

CONSTRUIREA COMPONENTELOR INTERFEȚEI CU UTILIZATORUL

În acest capitol:

- Prezentare generală a aplicației Form Designer din Access
- Crearea formularelor pentru aplicația TEB
- Crearea unui formular de tip meniu principal
- Rezumat

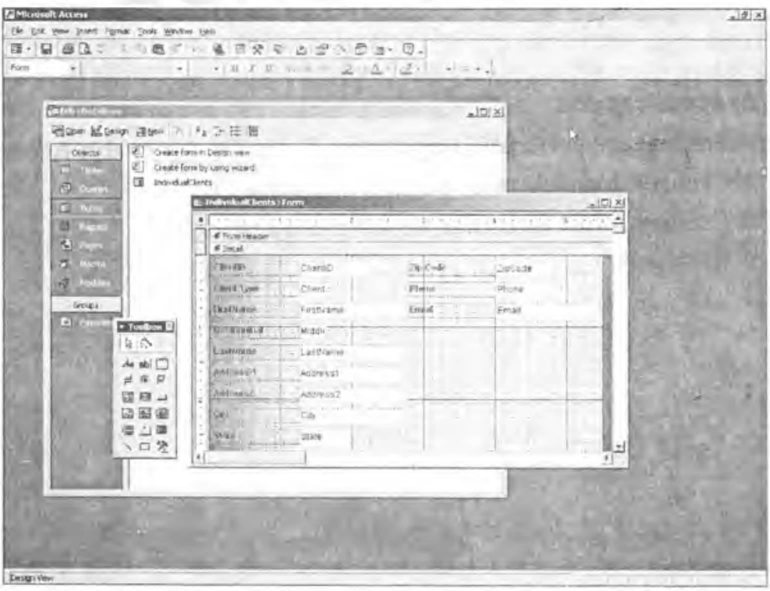
Pentru unii cititori, acest capitol ar putea reprezenta partea amuzantă a construirii aplicațiilor pentru baze de date. Construirea interfețelor cu utilizatorul, denumite de obicei în Access *formulare*, este o sarcină incredibil de ușoară. Dacă nu v-ați familiarizat cu scrierea codurilor de program, nu vă faceți griji! Puteți realiza 99 % din ceea ce trebuie să faceți fără să scrieți vreo linie de cod. Iar dacă trebuie să scrieți coduri, nu-i nimic! Ceea ce trebuie să scrieți și locul în care trebuie să scrieți vor fi clar explicate în acest capitol.

Acest capitol va aduce laolaltă mai multe concepte care au fost deja prezentate în carte. Mai precis, va trebui să revedeți câteva atribute ale câmpurilor și definiții ale interogărilor stocate în baza de date. La sfârșitul capitolului, veți înțelege bine cum pot fi combinate componentele bazei de date și ale formularelor pentru a oferi soluții complete.

Prezentare generală a aplicației Form Designer din Access

Cu mult timp în urmă, în capitolul 2, a fost prezentată pe scurt aplicația Form Designer din Access. Tot ce v-am spus atunci a fost că această aplicație există și că vom vorbi detaliat despre ea, mai târziu. În acest capitol vom respecta această promisiune. Figura 11.1 prezintă un formular numit IndividualClients.

FIGURA 11.1
Formularele, ca toate obiectele din Access, sunt stocate chiar în baza de date.



Aplicația Form Designer din Access are două componente principale: suprafața de proiectare și caseta de instrumente. Din fericire, cea mai mare parte din munca de creare a interfețelor cu utilizatorul poate fi efectuată folosind aplicații wizard. Din motive întemeiate, utilizarea acestor aplicații este evitată aici. Când trebuie să lucrați cu componentele esențiale ale bazei de date și să înțelegeți funcționarea generală a bazelor de date, trebuie să aflați ce se petrece în culise. Dacă lucrați cu formularele, aplicațiile wizard vă ajută să începeți, ocupându-se de părțile grele în locul dumneavoastră. În multe cazuri, dumneavoastră interveniți după aceea, pentru a ajusta anumite implementări.

Pe măsură ce veți construi formulare, veți vedea că procedura este aceeași. Cu alte cuvinte, veți urma aceiași pași de bază, și anume:

1. Lansați aplicația Form Wizard.
2. Alegeți un tabel sau o interogare pe care se bazează formularul.
3. Alegeți câmpurile care vor fi afișate în formular.
4. Alegeți o așezare a elementelor (*layout*) și un stil.
5. Salvați formularul.
6. Intrați în New Form Definition și faceți mici ajustări.

Crearea formularelor pentru aplicația TEB

Secțiunile următoare vă vor călăuzi prin procesul de construire a formularelor pentru a întreține datele despre clienți și angajați. În plus, veți crea și un formular de tip meniu principal care va fi utilizat pentru a lansa alte formulare. Vom realiza două formulare pentru clienți: unul pentru firme și altul pentru clienți persoane fizice. Pentru a construi aceste formulare, trebuie create două interogări. Ca să realizați formularul pentru angajați, vor fi necesare două căutări. Dacă vă amintiți, orice angajat aparține unui departament și îi este atribuită o clasificare. Poate vă amintiți și că au fost configurate diferite atribute ale câmpurilor pentru a facilita crearea formularelor. Nu vă temeți de necesitatea de a reciti pasaje anterioare ale cărții, deoarece întregul proces va fi recapitulat în acest capitol!

Formularele pentru clienți

Poate vă mai amintiți că există două tipuri de clienți: persoane fizice și firme. Fiecare tip este diferențiat folosind câmpul ClientType. Pentru clienții persoane fizice sunt completate câmpurile cu prenumele numele de familie și inițiala. Pentru clienții care sunt firme, este completat câmpul cu numele organizației. Când oferiți utilizatorilor o metodă de introducere a datelor, doriți să le fie cât mai ușor posibil. În acest caz, este logic să creați două interfețe diferite.

Cea mai simplă cale de a construi două interfețe diferite este să definiți două interogări: una pentru clienții persoane fizice, alta pentru clienții care sunt firme. Întrucât știți deja să construiți interogări, manual sau folosind aplicația de creare grafică, acest capitol vă va învăța cum să utilizați aplicația Query Wizard pentru a defini aceste interogări.

Interogarea IndividualClient

Figura 11.2 prezintă lansarea plicației Query Wizard. Motivul utilizării interogărilor în acest caz este faptul că un tabel are două roluri. Pentru a îndeplini aceste două roluri, este logic să-l divizați în două părți: clienții persoane fizice și clienții firme.

FIGURA 11.2

Aplicația Query Wizard poate fi lansată din fereastra principală a bazei de date.

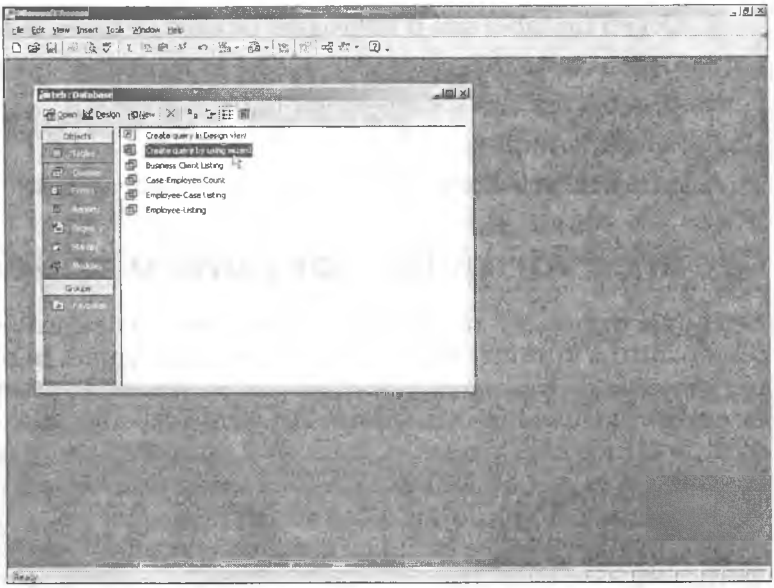


Figura 11.3 prezintă primul pas în definirea interogării, care presupune alegerea unui tabel sau a unei alte interogări. În acest caz, este utilizat tabelul Client. Întrucât această interogare va implica numai clienții persoane fizice, nu este necesar câmpul OrganizationName în setul de rezultate. De aceea, acest câmp a fost omis.

Figura 11.4 prezintă al doilea pas al aplicației Query Wizard, care implică denumirea interogării și specificarea dacă aceasta va fi executată sau va mai fi modificată înainte de execuție.

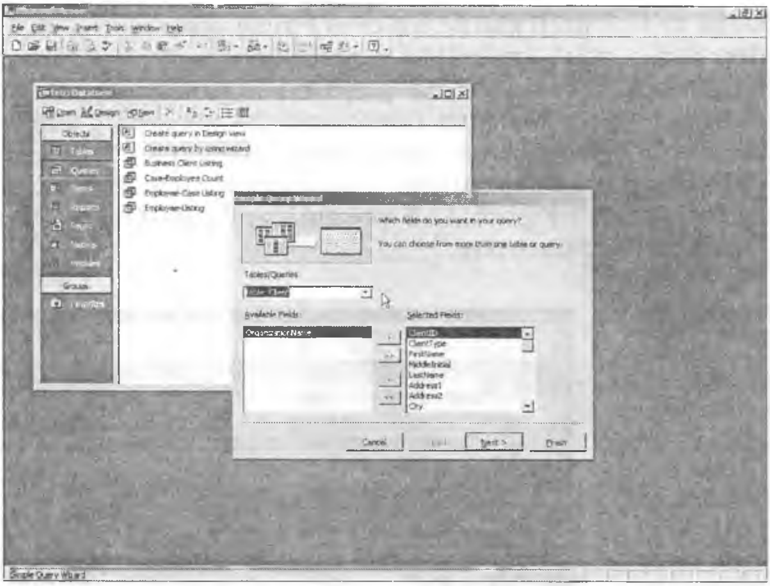


FIGURA 11.3

Pasul 1 al aplicației Query Wizard vă permite să definiți câmpurile care vor fi incluse în setul de rezultate.

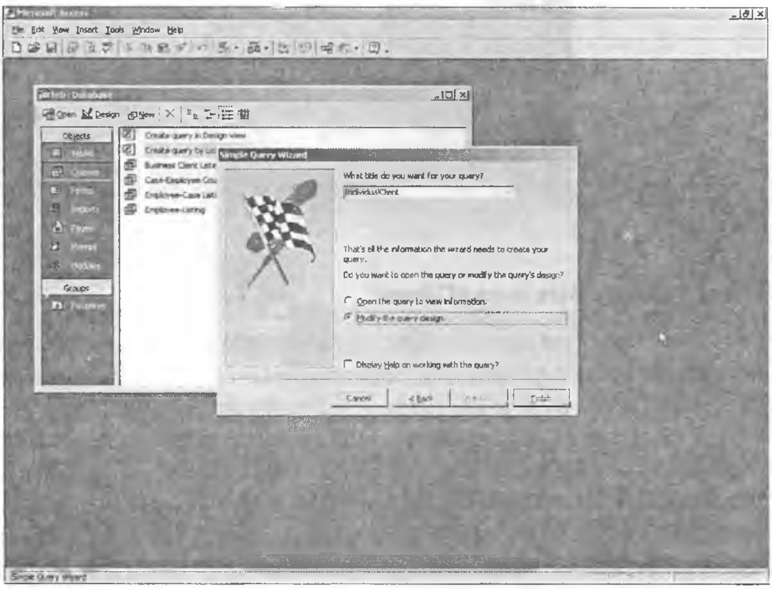


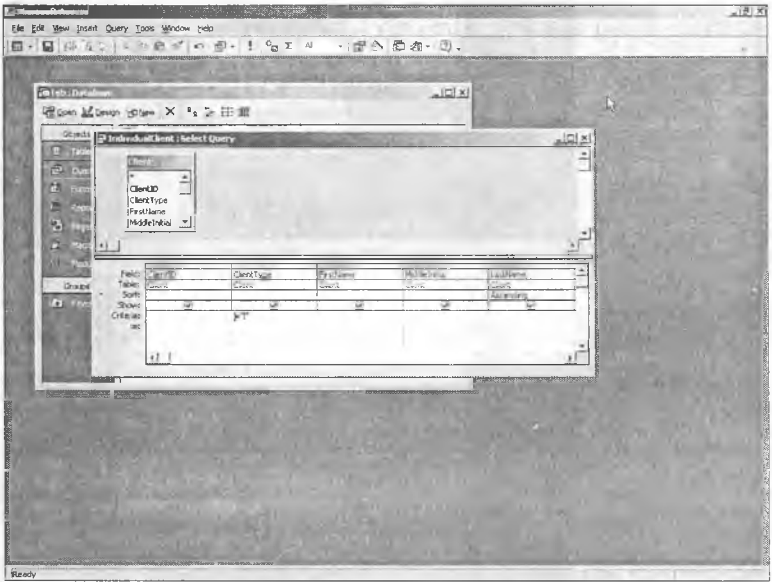
FIGURA 11.4

Pasul 2 al aplicației Query Wizard vă permite să furnizați un nume pentru interogare și să precizați dacă ea va fi executată direct sau va fi deschisă pentru a fi modificată înainte de salvare.

După ce executați clic pe butonul Finish, interogarea va fi salvată și va fi deschisă o nouă instanțiere a aplicației Query Designer. Există doi pași suplimentari care trebuie parcurși înainte de finalizarea interogării IndividualClient. Mai întâi, interogarea trebuie sortată după nume de familie și în ordine ascendentă. Apoi, trebuie introdus în câmpul ClientType un criteriu pentru a se returna numai înregistrările care îndeplinesc condiția ClientType="I". Figura 11.5 prezintă aplicația Query Designer și aceste modificări suplimentare.

FIGURA 11.5

La interogarea IndividualClient, creată folosind aplicația Query Wizard, au fost adăugate o ordine de sortare și un criteriu.



Interogarea BusinessClient

Pașii necesari pentru realizarea interogării BusinessClient sunt aproape aceiași cu cei necesari pentru crearea interogării IndividualClient. O diferență este faptul că în loc de prenume, numele de familie și inițială apare un câmp cu numele organizației. Mai departe, ordinea de sortare a elementelor din acest câmp este ascendentă. Altă diferență se referă la criteriu. În interogarea BusinessClient, vor fi returnate în setul de rezultate numai înregistrările care îndeplinesc condiția ClientType = "B". Figura 11.6 prezintă interogarea BusinessClient în aplicația de proiectare a interogărilor.

Figura 11.7 prezintă ambele interogări în acțiune. După ce am terminat cu interogările, poate începe munca de adunare a clienților persoane fizice și clienților firme.

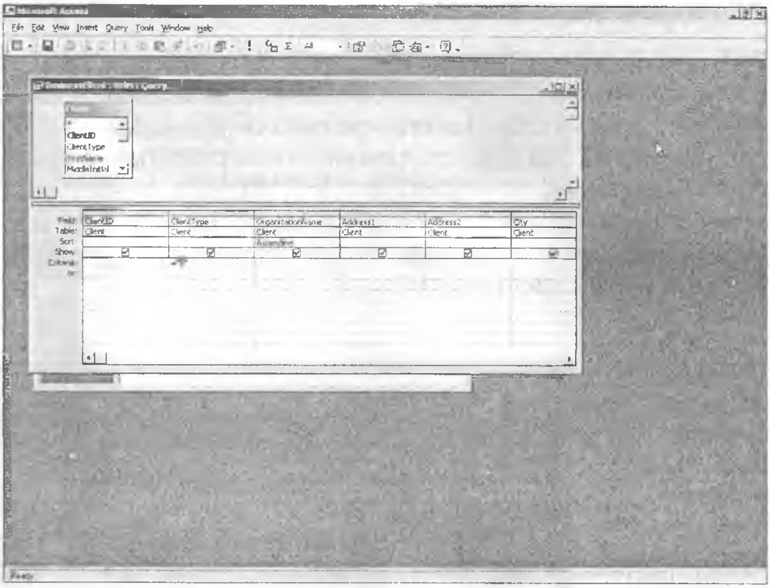


FIGURA 11.6
Interogarea BusinessClient, completată cu o sortare și un criteriu în Query Designer.

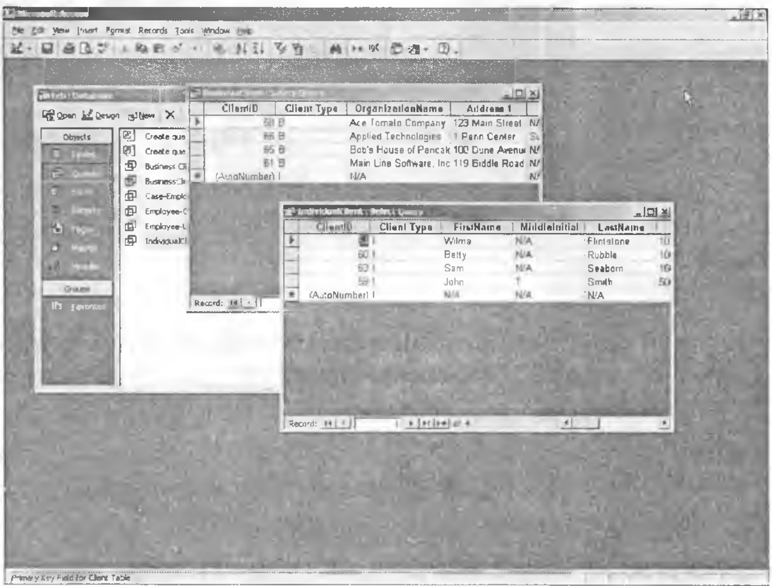


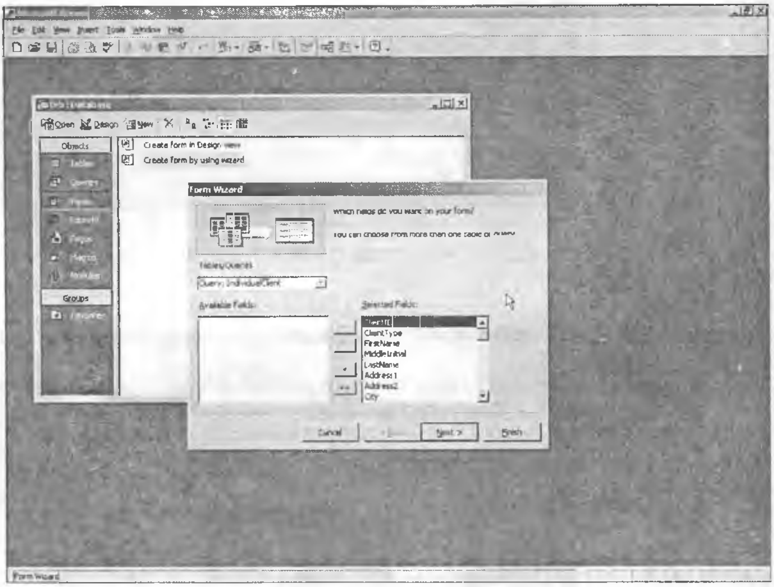
FIGURA 11.7
Interogările care returnează clienții persoane fizice și clienții care sunt firme oferă două moduri de a privi tabelul Client.

Formularul IndividualClient

Dacă nu ați fi avut la dispoziție aplicații wizard care să se ocupe de partea grea a construirii formularelor, această muncă ar consuma mult timp. Deși ați fi putut să vă bazați numai pe ceea ce vă oferă aplicația wizard, pentru a obține o interfață mai funcțională și mai prietenoasă față de utilizator veți dori să modificați formularul după ce îl creați. Vestea cea bună este că și pentru multe dintre sarcinile pe care le veți executa după crearea formularului există aplicații wizard! Figura 11.8 prezintă primul pas de utilizare a aplicației Form Wizard.

FIGURA 11.8

Pasul 1 al aplicației FormWizard vă permite să alegeți un tabel sau o interogare și câmpurile corespunzătoare pe care se va baza formularul.



În acest caz, la baza formularului IndividualClient stă interogarea IndividualClient. Pentru toate aceste formulare, vom alege să utilizăm toate câmpurile.

Figura 11.9 prezintă pasul 2. În acest pas, puteți alege așezarea elementelor (*layout*) în formular. În acest caz, va fi selectată opțiunea implicită de așezare pe coloane.

În pasul 3, prezentat în figura 11.10, puteți alege un stil. Access conține mai multe stiluri. În aceste exemple, va fi utilizat stilul Blends.

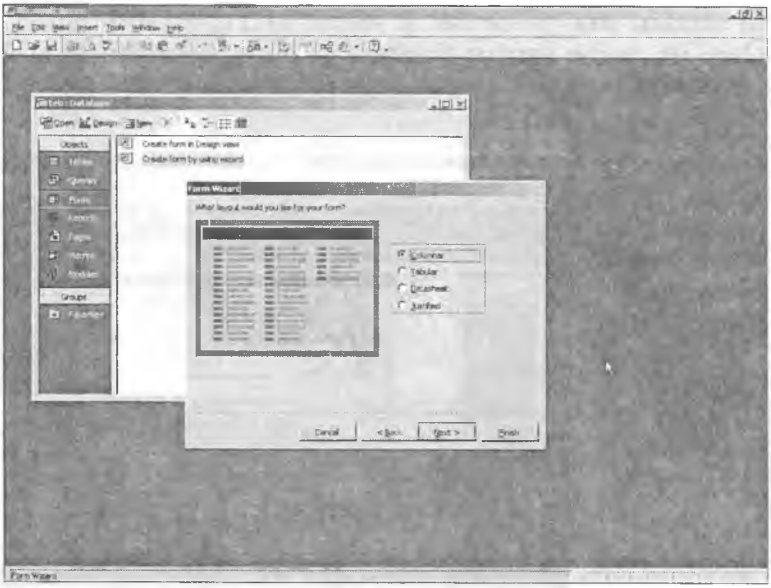


FIGURA 11.9
Pasul 2 al aplicației Form Wizard vă permite să alegeți o așezare a elementelor din formular.

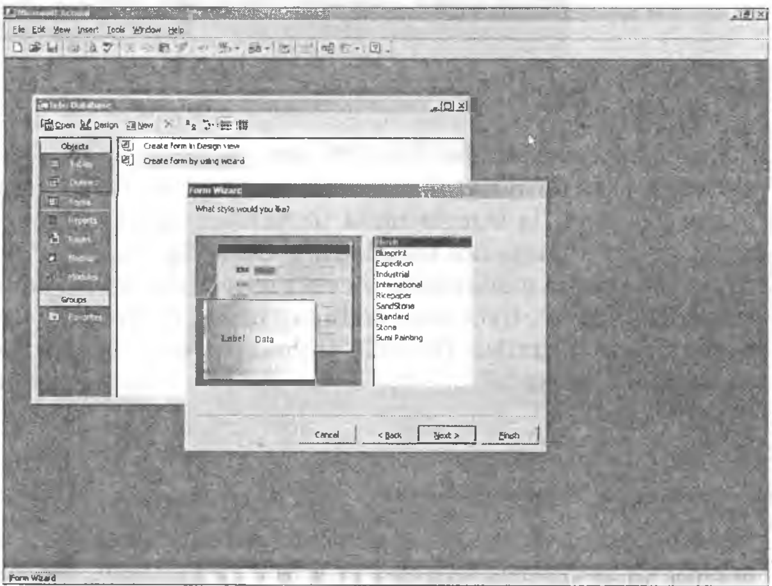


FIGURA 11.10
Pasul 3 al aplicației Form Wizard vă permite să alegeți un stil pentru formular.

În ultimul pas, prezentat în figura 11.11, ca și în Query Wizard, puteți alege un nume pentru formular și puteți specifica dacă formularul va fi rulat direct sau va mai fi modificat înainte de salvare.

FIGURA 11.11
Formularul BusinessClient se bazează pe interogarea BusinessClient.

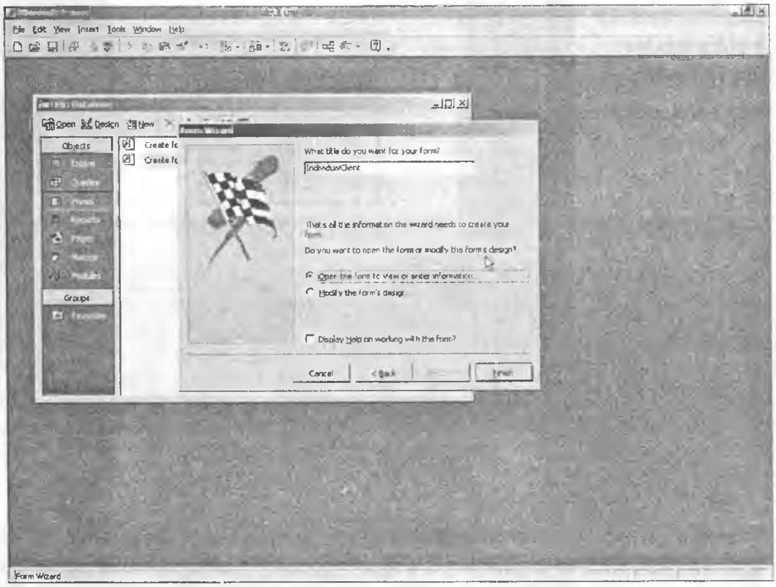


Figura 11.12 prezintă produsul finalizat. Folosind bara de instrumente din partea de jos a formularului, puteți trece la prima înregistrare, la înregistrarea precedentă, la înregistrarea următoare sau la ultima înregistrare. În plus, puteți adăuga noi înregistrări. Ce vă lipsește este posibilitatea de a șterge sau anula modificările. Acest lucru se datorează faptului că, după ce creați formularul, trebuie să adăugați un set de butoane care adaugă funcționalitate în formular. Detaliile privind procesul de adăugare a butoanelor de comandă personalizate vor fi discutate mai târziu, în cursul acestei secțiuni.

Formularul BusinessClient

Pașii necesari pentru crearea formularului BusinessClient sunt aproape aceiași ca la crearea formularului IndividualClient. Singura excepție este faptul că, în acest caz, formularul se va baza pe interogarea BusinessClient. Figura 11.13 prezintă formularul BusinessClient.

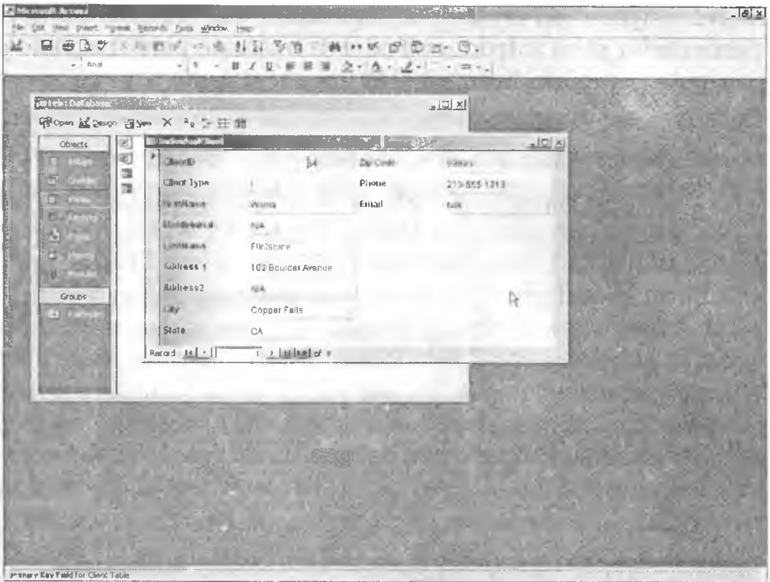


FIGURA 11.12

Formularul creat de aplicația wizard oferă capacitatea de a naviga prin înregistrări și posibilitatea de a crea noi înregistrări.

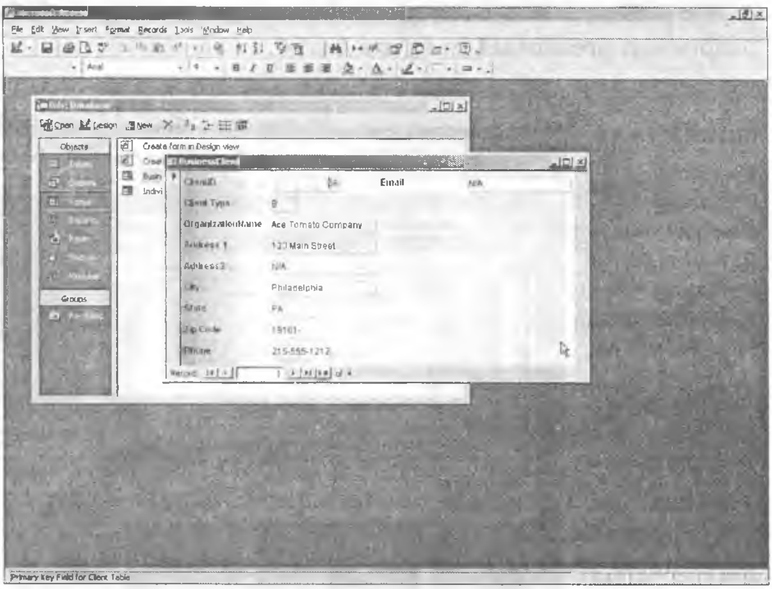


FIGURA 11.13

Formularul BusinessClient este bazat pe interogarea BusinessClient.

Acum, după ce ați creat două formulare, este timpul să modificați așezarea elementelor și să introduceți câteva butoane personalizate pentru comenzi pentru a completa funcționalitatea acestora.

Modificarea așezării elementelor din formulare

Deși aplicația wizard face foarte multe în locul dumneavoastră, nu poate face treaba în proporție de 100 %. Figura 11.4 prezintă așezarea modificată a formularului IndividualClient. Așezarea modificată are un aspect mai echilibrat și mai profesional. Schimbările nu se referă numai la poziție. Mai întâi, a fost șters câmpul ClientID. Nu este necesar ca utilizatorul să vadă acest câmp, deoarece el este folosit de baza de date. În al doilea rând, câmpul ClientType a fost mutat în partea de jos a formularului, iar proprietatea visible a fost setată pe No. Nu uitați că în fiecare formular pentru clienți fiecare client trebuie să aibă același tip de valoare. De aceea, nu este nevoie ca utilizatorul să vadă acest lucru. Totuși, câmpul nu oferă servicii valoroase pentru înregistrările noi. Veți vedea de ce este important acest câmp în secțiunea următoare. În sfârșit, unele dintre etichetele cu text utilizate pentru a descrie câmpurile de intrare au fost fie modificate, fie pur și simplu șterse. Pentru a șterge un element, trebuie doar să îl selectați cu mouse-ul și apăsați tasta Delete. Pentru a modifica textul unei etichete, executați clic pe ea pentru a o selecta și modificați proprietatea Caption în caseta de proprietăți prezentată în figura 11.14.

FIGURA 11.14

Versiunea modificată a formularului IndividualClient are un aspect mai echilibrat și mai profesional.

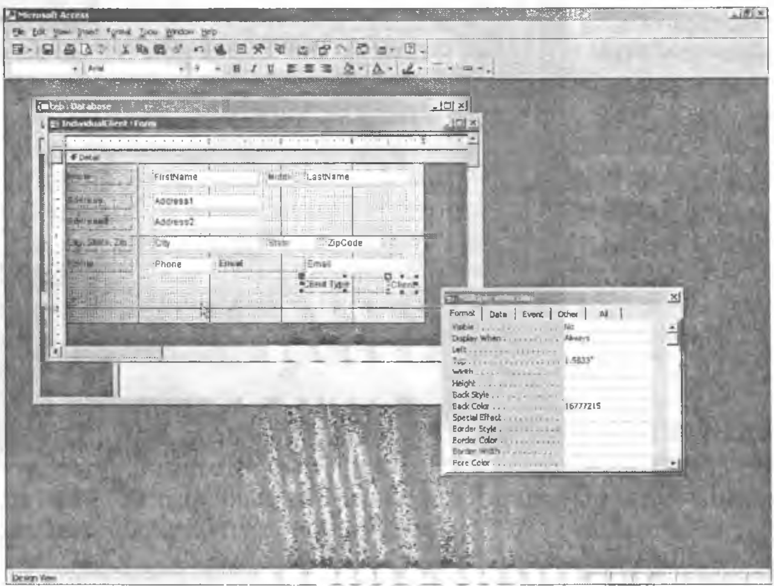
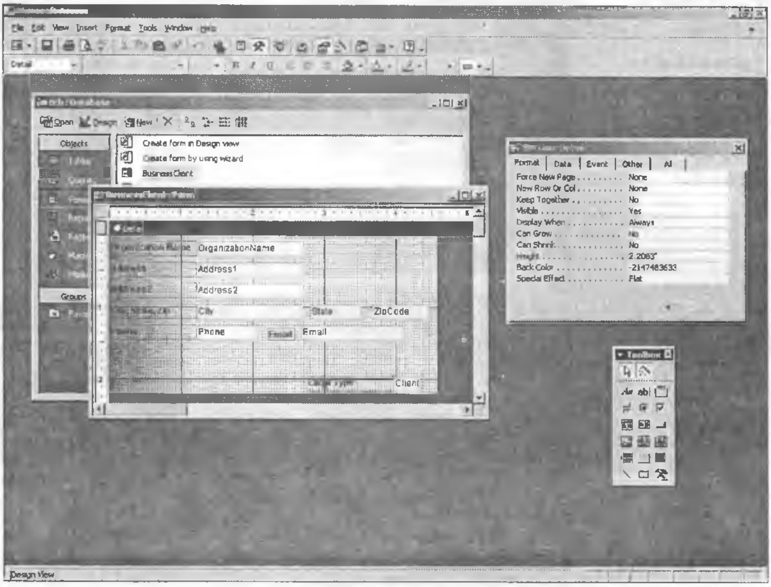


Figura 11.15 prezintă aceleași modificări, dar aplicate formularului BusinessClient.

FIGURA 11.15

Versiunea modificată a formularului BusinessClient are un aspect mai echilibrat și mai profesionist.



Introducerea butoanelor de comandă pentru navigarea prin înregistrări

Cele două formulare pentru clienți sunt aproape complete. Ultimul pas presupune introducerea butoanelor personalizate care facilitează navigarea prin înregistrări, introducerea și ștergerea înregistrărilor. În plus, va fi introdusă capacitatea de a salva și anula modificările, precum și cea de a găsi înregistrările pe baza unui criteriu de căutare. Din fericire, există aplicații wizard care pot îndeplini aceste sarcini și practic, nu este necesar să scrieți vreun cod. Așa cum veți vedea, va fi necesară o linie de cod în butonul Add.

Figura 11.16 prezintă formularul IndividualClient în modul Design. Caseta de instrumente Form Design, prezentată și ea în figura 11.16, conține o varietate de controale care pot fi introduse în formular. Aplicația Form Wizard a utilizat aceleași controale când au fost create primele versiuni ale formularelor.

Pentru a introduce un buton în formular, trebuie doar să executați clic pe pictograma butonului de comandă din caseta de instrumente. Figura 11.17 prezintă cum își schimbă mouse-ul forma.

FIGURA 11.16

Caseta de instrumente din Form Design conține o varietate de controale care pot fi introduse într-un formular.

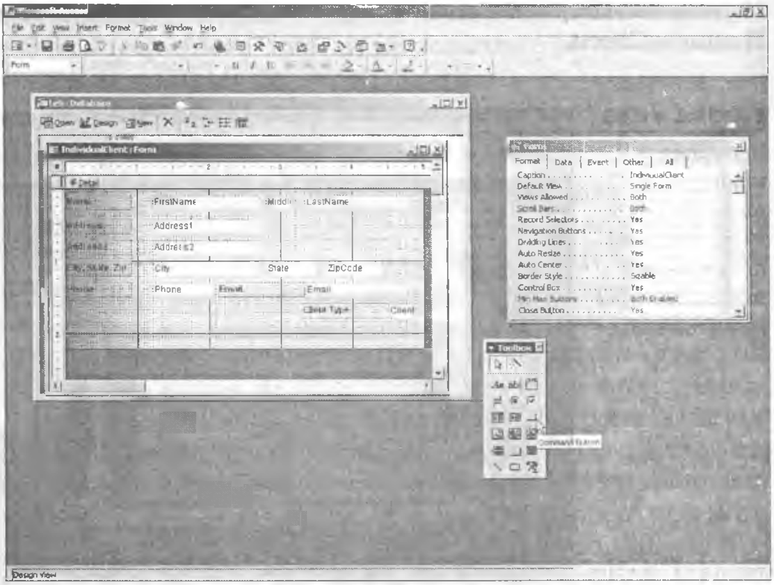
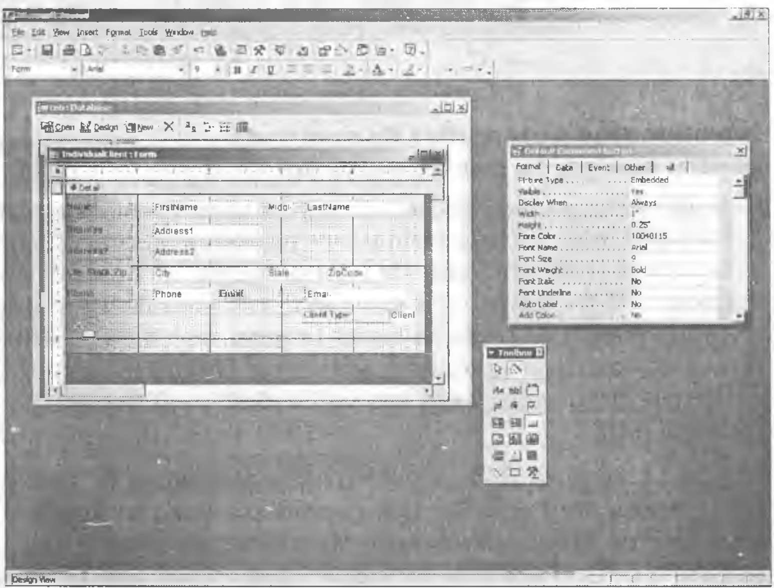


FIGURA 11.17

Dacă executați clic pe un element din caseta de instrumente, pointerul mouse-ului își schimbă înfățișarea, ajutându-vă să plasați obiectul pe suprafața de proiectare a formularului.



După ce selectați un loc din formular în care doriți să apară obiectul, executați un clic. Figura 11.18 prezintă primul pas al aplicației Command Button Wizard, care apare când obiectul este plasat în formular.

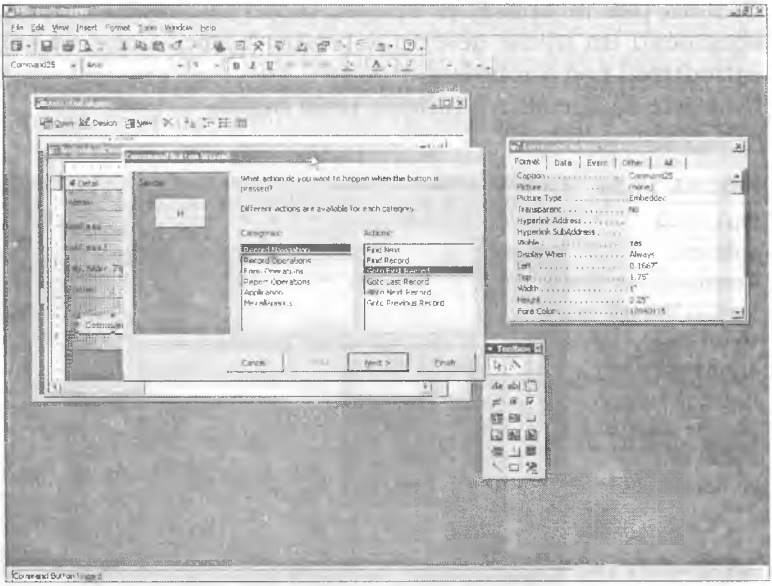


FIGURA 11.18

Primul pas al aplicației Command Button Wizard vă invită să selectați tipul de operație executată când se execută clic pe buton.

După ce selectați o operație, pasul 2, prezentat în figura 11.19, vă invită să selectați butonul care va apărea.

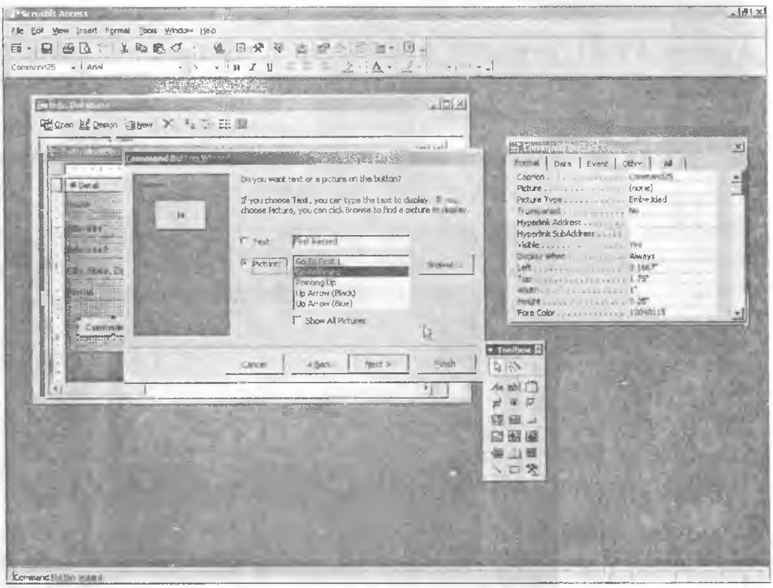


FIGURA 11.19

Pasul 2 al aplicației Command Button Wizard vă invită să selectați butonul care va apărea.

Ultimul pas al aplicației wizard, prezentat în figura 11.20, vă invită să specificați un nume descriptiv pentru buton. Când va sosi momentul să introduceți cod suplimentar pentru butonul Add, veți vedea cât de important este să specificați numele unui buton.

FIGURA 11.20
Pasul 3 al aplicației Command Button Wizard vă invită să specificați un nume descriptiv pentru obiectul buton de comandă.

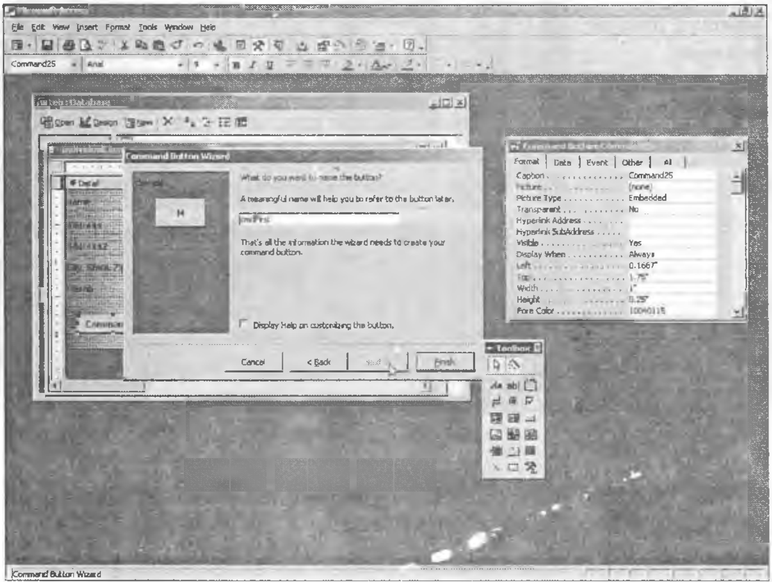
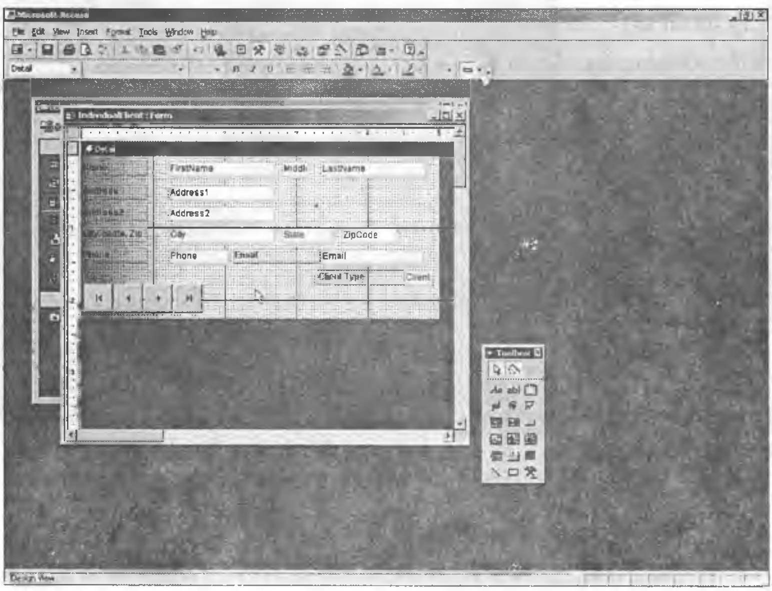


Figura 11.21 prezintă înfățișarea formularului IndividualClient după adăugarea celorlalte butoane de navigare.

FIGURA 11.21
Formularul IndividualClient cu butoanele de navigare personalizate.



După crearea butoanelor de navigare, pasul următor presupune introducerea capacităților care permit utilizatorului să creeze și să șteargă înregistrări, să salveze și să anuleze modificările asupra înregistrărilor, iar în final să caute o anumită înregistrare după un criteriu de căutare.

Introducerea capacităților suplimentare în formularul IndividualClient

Introducerea funcției de creare și ștergere a înregistrărilor, salvare și anulare a modificărilor asupra înregistrărilor, precum și a funcției de căutare, este la fel de ușoară ca și introducerea unui alt tip de buton de comandă. Funcția de căutare poate fi găsită în secțiunea Record Navigation din aplicația Command Button Wizard. Funcția suplimentară în această secțiune se găsește în secțiunea Record Operations. Figura 11.22 prezintă primul pas pentru introducerea unei noi înregistrări.

FIGURA 11.22
Secțiunea Record Operations a aplicației wizard oferă posibilitatea de a crea, adăuga, șterge, salva și anula înregistrările într-un formular.

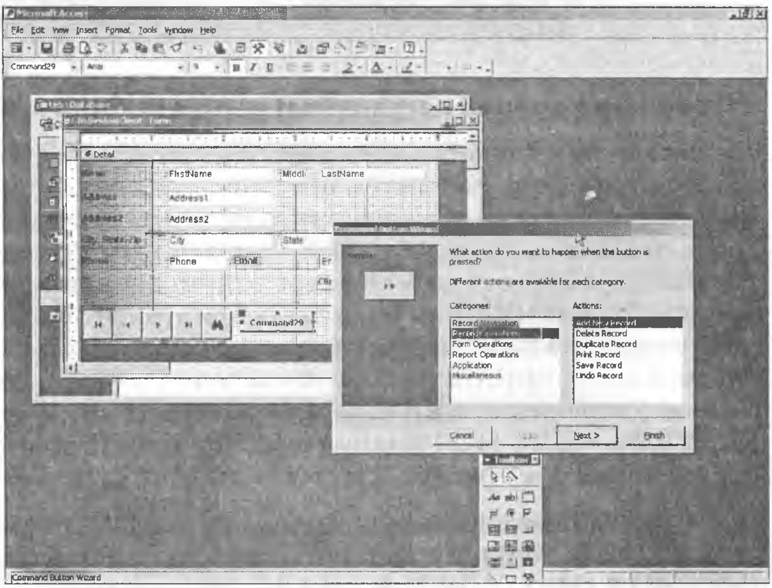


Figura 11.23 prezintă formularul IndividualClient, cu funcțiile suplimentare pe care le necesită un formular obișnuit pentru introducerea datelor.

După introducerea tuturor butoanelor mai este necesar un element de personalizare. Dacă vă mai amintiți, deși am promis că nu prea va fi nevoie să scrieți cod, va trebui totuși să introduceți o linie de cod pentru butonul Add (de introducere a înregistrărilor). Această sarcină este îndeplinită în secțiunea următoare.

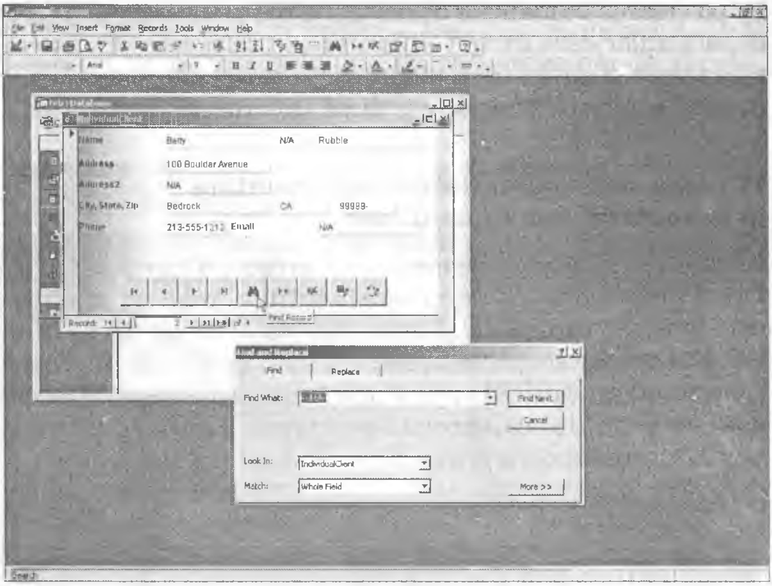


FIGURA 11.23
Formularul IndividualClient, completat cu funcțiile tipice necesare unui formular de introducere a datelor.

Introducerea codului personalizat pentru butonul Add

Controalele dintr-un formular răspund unei varietăți de evenimente. Evenimentele tipice sunt executarea unui clic, executarea unui dublu clic și apăsarea unei taste, pentru a enumera numai câteva. Dacă este apăsat un buton, are loc evenimentul On Click. Programarea în funcție de evenimente este bazată pe atașarea codurilor pentru diverse evenimente ale sistemului. Aplicația Command Button Wizard se ocupă în locul dumneavoastră de cea mai mare parte a codului. Există însă câteva cazuri când trebuie să interveniți după aceea, pentru a mări funcționalitatea butonului respectiv. În acest caz, codul suplimentar are rolul de a da câmpului ClientType valoarea corespunzătoare. Figura 11.24 prezintă modul de acces la procedura evenimentului On Click din fereastra Property.

Lângă procedura evenimentului On Click din fereastra Property, prezentată în figura 11.24, veți observa un mic buton cu o elipsă pe el. După ce vă asigurați că este selectat butonul Add, dacă executați clic pe butonul cu elipsa, va apărea o fereastră Visual Basic Code Editing (pentru editarea codului în Visual Basic). Această fereastră și codul suplimentar care trebuie introdus sunt prezentate în figura 11.25.

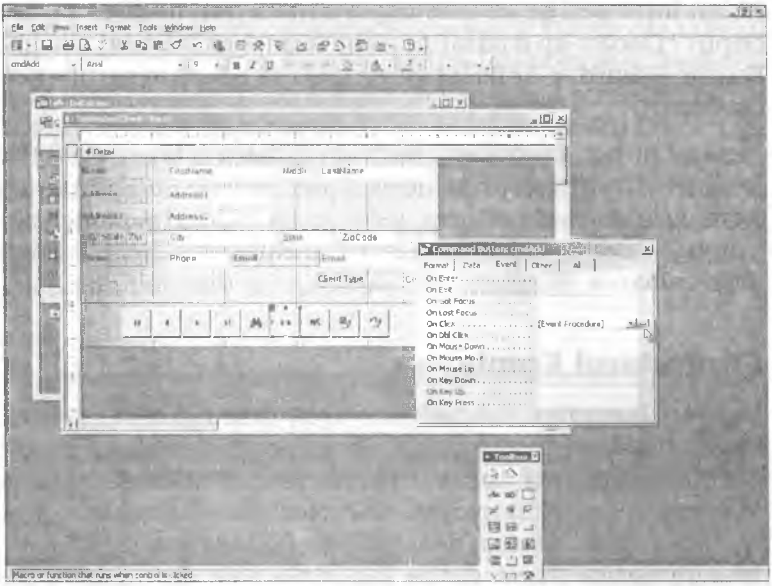


FIGURA 11.24

Procedurile pentru evenimente corespunzătoare controalelor pot fi accesate din fereastra Event a ferestrei Property.

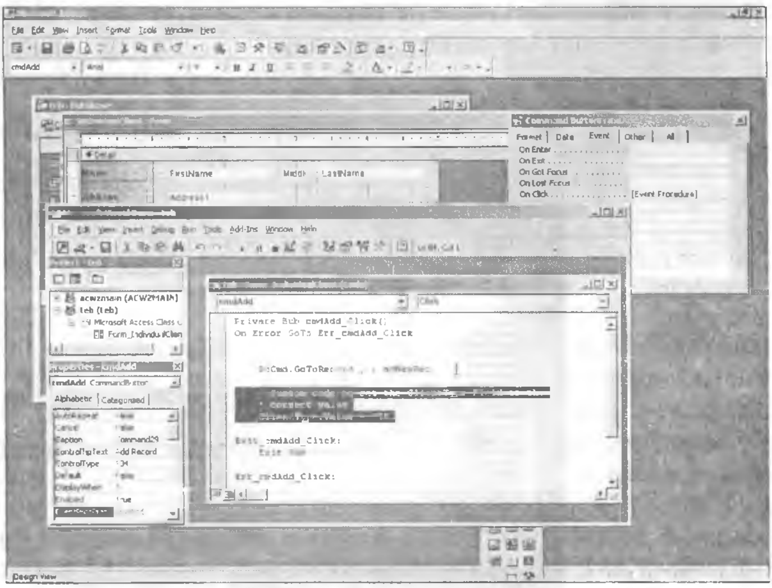


FIGURA 11.25

Codul suplimentar, introdus prin procedura cmdAdd_Click(), atribuie câmpului ClientType valoarea corespunzătoare.

Acum înțelegeți de ce este important să dați fiecărui buton un nume descriptiv? Dacă v-ați fi bazat pe numele implicit furnizat de aplicația Command Button Wizard, v-ar fi fost foarte greu să diferențiați diferitele coduri pentru proceduri.

În acest punct, formularul IndividualClient este complet. Pentru formularul BusinessClient veți urma aceiași pași pe care i-ați parcurs pentru a pregăti formularul IndividualClient, cu excepția codului personalizat adăugat pentru butonul Add. În cazul formularului BusinessClient, câmpul Clienttype va avea valoarea "B".

Formularul Employee

Ultimul formular despre care vom discuta în acest capitol stochează înregistrările despre angajați. Spre deosebire de formularele pentru clienți, formularul pentru angajați nu necesită o interogare, ci va acționa direct asupra tabelului Employee. Caracteristica interesantă a acestui formular nu este complexitatea, deoarece, de fapt, acesta este foarte simplu, ci felul în care derivă unele dintre însușirile sale din metadatele tabelului Employee.

Poate vă amintiți că într-o bază de date din Access pot fi stocate variate caracteristici și metadate de căutare. Figura 11.26 prezintă metadatele de căutare pentru câmpul DepartmentID.

FIGURA 11.26
Controalele afișate și metadatele de căutare pot fi stocate în metadatele câmpului.

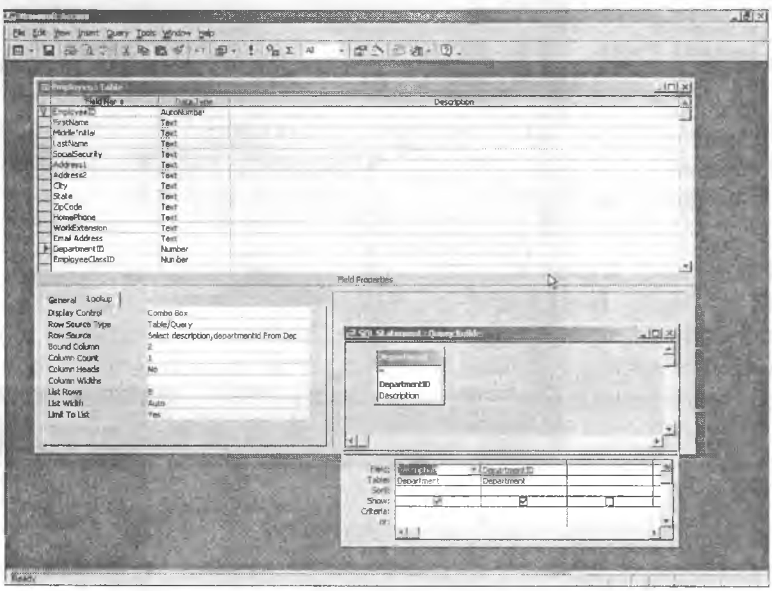
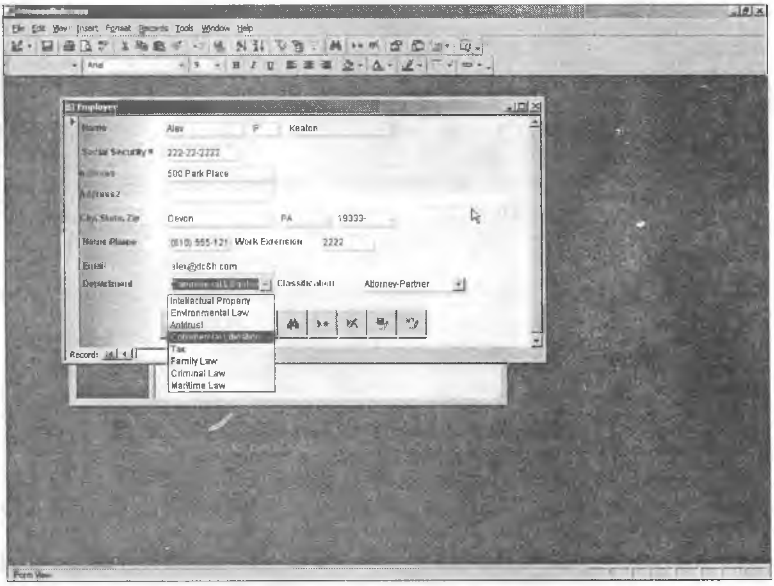


Figura 11.26 arată că pentru a afișa datele de căutare din tabelul Department este utilizat un control de tip casetă combinată derulantă. Datele din tabelul Department se găsesc în proprietatea RowSource din tab-ul lookup. Probabil vă amintiți că între tabelele Department și Employee există o relație unu la mulți. Cheia externă a câpului DepartmentID, preluată în tabelul Employee, corespunde cheii primare din tabelul Department. În acest mod, elementele din caseta combinată sunt sincronizate corespunzător cu înregistrările din tabelul Employee. Aplicația Form Wizard utilizează informațiile din metadate când construiește formularul Employee. Figura 11.27 prezintă formularul Employee, completat cu butoane de comandă personalizate.

FIGURA 11.27
Multe dintre caracteristicile și controalele unui formular derivă din metadatele bazei de date.



După ce ați creat câteva formulare pentru date, veți începe să vedeți foarte multe elemente comune în procesul de construire a formularelor. Cu alte cuvinte, după ce construiți câteva formulare, formularele încep să semene unul cu altul! Singurul tip de formular despre care nu am discutat până acum este cel care stochează datele din două sau mai multe tabele simultan. Aceste tipuri de formulare au un nivel de complexitate care depășește scopurile acestei cărți. Pentru a vedea un astfel de formular, puteți deschide baza de date Northwind Traders, care vă este oferită împreună cu programul Access.

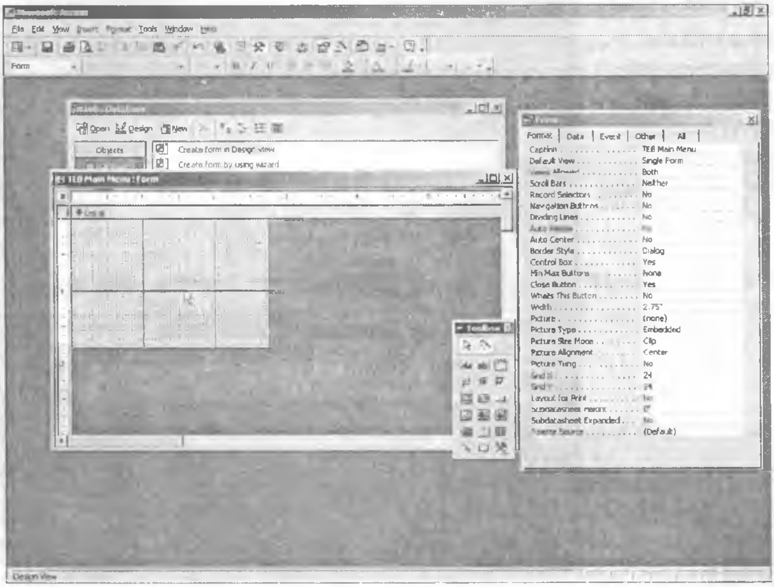
Secțiunea următoare prezintă construirea unui formular care nu este bazat pe date. Mai precis, veți afla cum puteți construi un formular de tip meniu principal. În plus, veți învăța cum puteți manevra diferitele proprietăți ale formularului, pe care veți dori să le utilizați atât în formularele bazate pe date cât și în formularele care nu sunt bazate pe date.

Crearea unui formular de tip meniu principal

Figura 11.28 prezintă o sesiune de creare a unui formular, inițiată prin deschiderea directă a vizualizării Design, nu a aplicației wizard. Deoarece acest formular nu se bazează pe date, nu sunt necesare butoanele de navigare prin înregistrări care apar în mod normal în partea de jos a formularului. Există alte atribute pe care poate doriți să le manipulați, cum ar fi faptul că este sau nu activat butonul min/max, tipul de chenar și legenda (explicația) formularului, pentru a numi numai câteva.

FIGURA 11.28

Fișa de proprietăți a formularului vă permite să manipulați diferite atribute ale formularului.



După configurarea formularului, puteți să introduceți butoanele de comandă care vor lansa formularele create în acest capitol. Vestea bună este că aplicația Command Button Wizard face aproape toată munca în locul dumneavoastră. Figura 11.29 prezintă primul pas pentru introducerea unui buton care lansează un formular.

Figura 11.30 prezintă al doilea pas al aplicației wizard, care vă solicită numele formularului care va fi lansat.

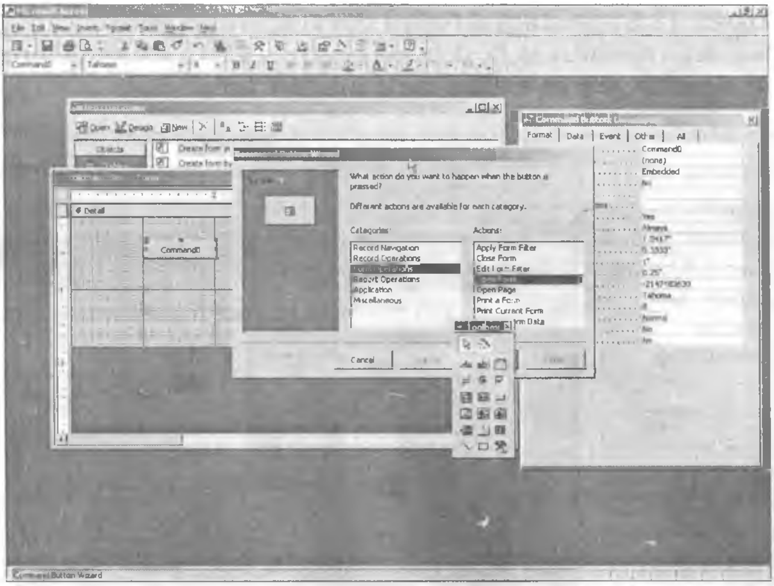


FIGURA 11.29

În procesul de introducere a unui buton care lansează un formular este utilizată aplicația Command Button Wizard.

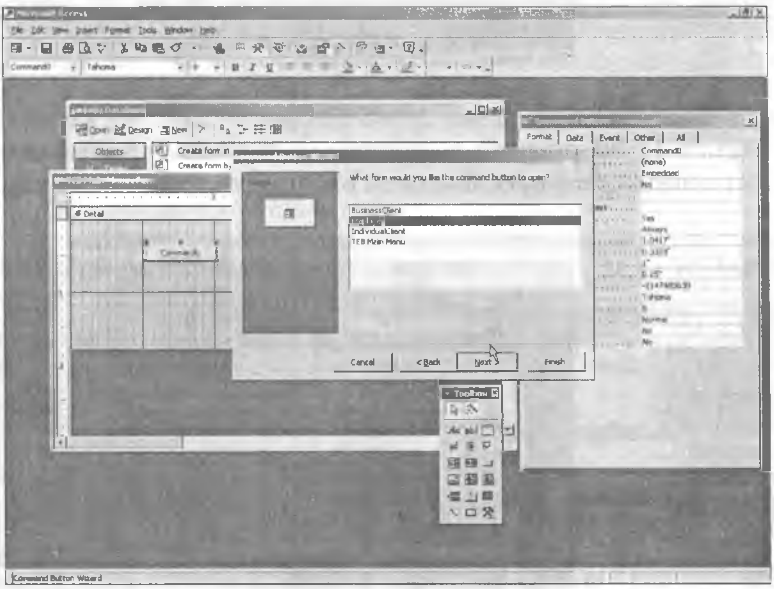


FIGURA 11.30

Al doilea pas al aplicației wizard solicită utilizatorului numele formularului care va fi lansat.

Figura 11.31 prezintă al treilea pas al aplicației wizard, care oferă posibilitatea ca formularul să fie deschis pentru a găsi o anumită înregistrare sau să fie deschis pentru a afișa toate înregistrările.

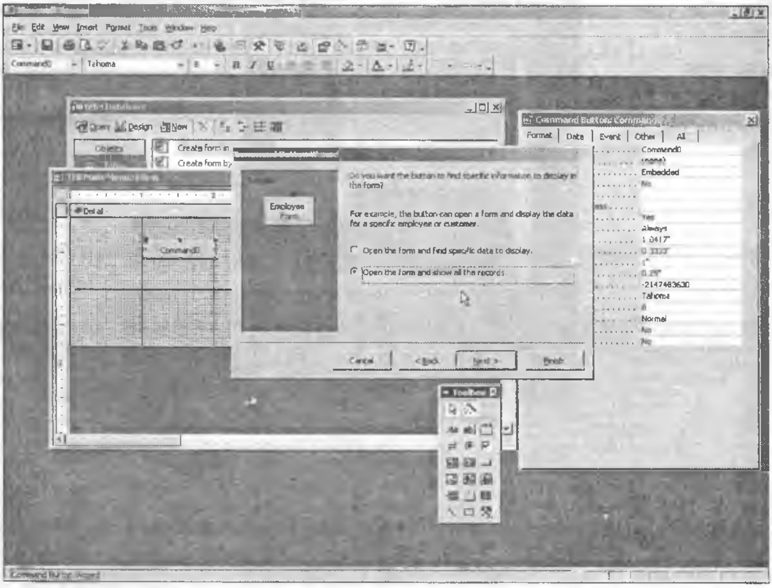


FIGURA 11.31

Al treilea pas al aplicației wizard specifică modul în care va fi deschis formularul.

Figura 11.32 prezintă al patrulea pas al aplicației wizard, care vă oferă capacitatea de a specifica dacă butonul va conține text sau o imagine.

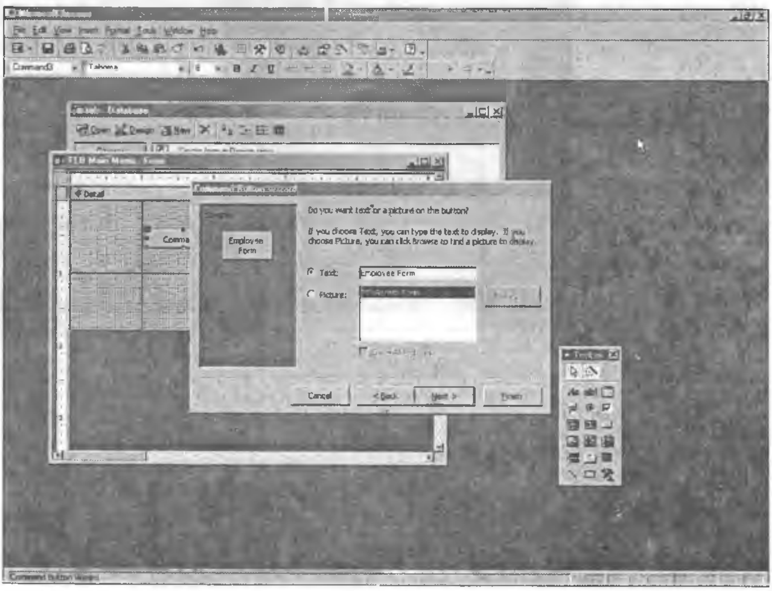


FIGURA 11.32

Al patrulea pas al aplicației wizard specifică aspectul butonului.

Figura 11.33 prezintă al cincilea pas al aplicației wizard, care oferă posibilitatea de a introduce un nume descriptiv pentru buton.

FIGURA 11.33
Al cincilea pas al aplicației wizard oferă posibilitatea de a specifica un nume descriptiv pentru buton.

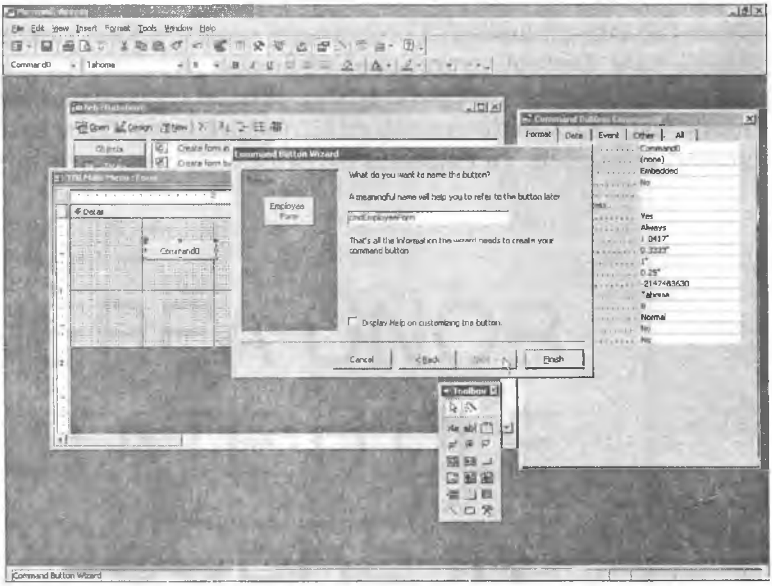
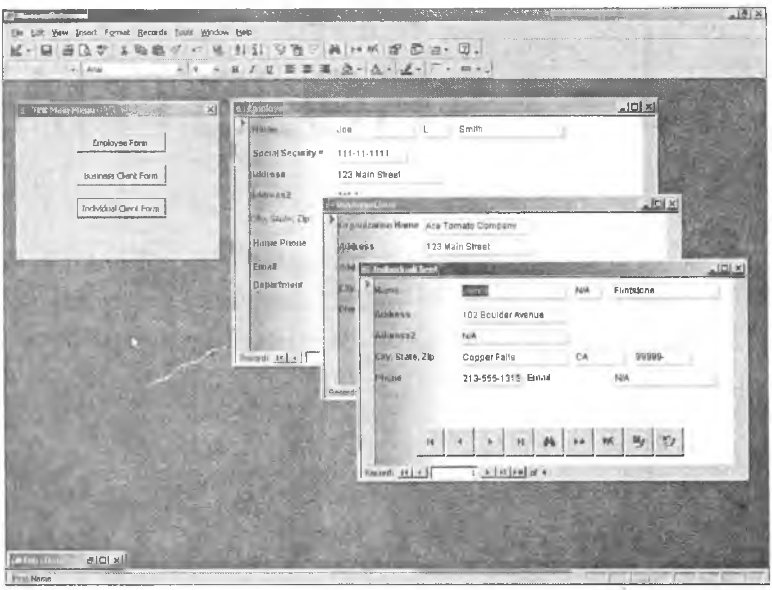


Figura 11.34 prezintă formularul complet de tip meniu, precum și cele trei formulare create în acest capitol.

FIGURA 11.34
Formularul complet de tip meniu principal și cele trei formulare create în acest capitol.



Rezumat

În acest capitol ați văzut cât de simplă este construirea unui formular în Access. Întrucât aplicațiile wizard pot face aproape toată munca, nu trebuie să excelați într-un limbaj de programare pentru a construi o aplicație pentru baze de date. De fapt, ar trebui să vă fie clar că sunteți pe calea cea bună în construirea unei aplicații pentru baze de date! Ultimul pas al acestei călătorii inițiale este să oferiți utilizatorului posibilitatea de a extrage informații folosind rapoartele. În capitolul următor vom discuta despre rapoarte.



12

OBȚINEREA UNOR REZULTATE UTILE: INTRODUCERE ÎN CREAREA ȘI PREZENTAREA RAPOARTELOR

În acest capitol:

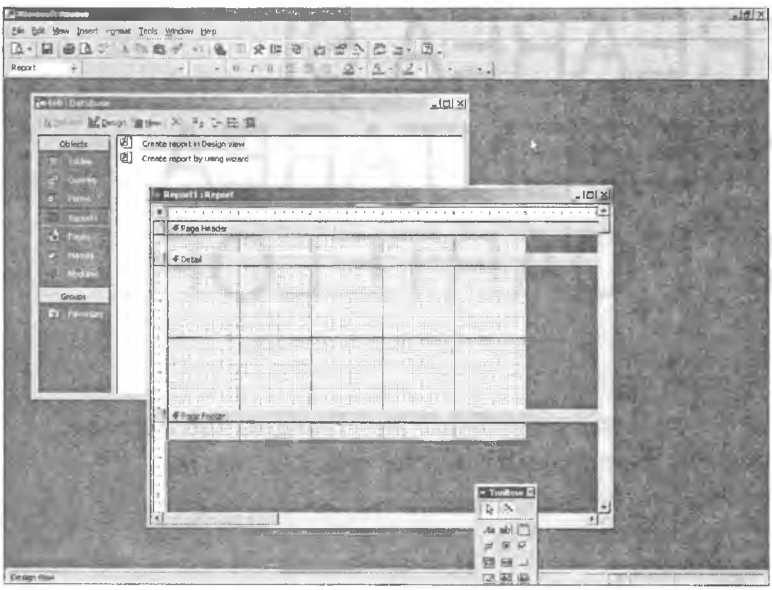
- Prezentare a aplicației Report Designer din Access
- Crearea rapoartelor pentru aplicația TEB
- Introducerea elementelor rapoartelor în formularul de tip meniu principal
- Rezumat

În această carte am afirmat că valoarea unei baze de date constă în calitatea datelor pe care le conține și a informațiilor pe care le poate oferi. Până acum, în general, ați lucrat cu elemente de date și înregistrări individuale. Singura excepție a apărut când am prezentat interogările și limbajul SQL (Structured Query Language). Folosind interogările, puteți colecta și agrega datele. Procesul de rulare a interogărilor este adesea denumit transformarea datelor în informații. Se menține însă problema modului în care sunt prezentate informațiile pentru utilizator. Ea este rezolvată prin existența posibilității de raportare a datelor din aplicație. Rapoartele sunt, probabil, caracteristica cea mai importantă a unei baze de date, precedate numai de calitatea bazei de date și a datelor propriu-zise. În acest capitol vom prezenta aplicația Report Designer din Access și tehnicile de bază pe care trebuie să le înțelegeți pentru a crea rapoarte simple în Access.

Prezentare generală a aplicației Report Designer din Access

Ca și în cazul formularelor, „cu mult timp în urmă", în capitolul 2, am prezentat pe scurt aplicația Report Designer din Access. Acest capitol îndeplinește promisiunea de a vă arăta cum să utilizați această aplicație.

FIGURA 12.1
Aplicația Report Designer din Access.



Toate aplicațiile de proiectare a rapoartelor funcționează, în linii generale, asemănător. Majoritatea au o abordare de tip „benzi" (*banded*). Cu alte cuvinte, când proiectați un raport, aveți la dispoziție în pagină o bandă pentru antet (*header*), pentru subsol (*footer*) și benzi de detalii. Figura 12.1 prezintă aceste trei benzi. Numele benzilor reflectă cu acuratețe conținutul lor. Antetul paginii conține de obicei titlul raportului și etichete care descriu conținutul acestuia. Subsolul paginii conține de obicei numărul paginii și elemente de date. Banda de detalii, așa cum sugerează numele ei, conține raportul detaliat. După cum veți vedea, banda de detalii poate fi împărțită în alte benzi de detalii. Nivelul de divizare a unei benzi de detalii depinde de modul de grupare a datelor. Vom discuta în acest capitol și despre modul de grupare într-un raport. Figura 12.1 prezintă și caseta de instrumente pentru controale, aceeași cu care ați lucrat când ați proiectat și construit formulare. Din fericire, fiindcă Access este plin de aplicații wizard, în general nu va fi nevoie să intrați în detaliile neplăcute ale construirii unui raport de la zero. În acest capitol vom utiliza foarte mult aplicația Report Wizard.

Crearea rapoartelor pentru aplicația TEB

Acest capitol vă va conduce prin procesul creării unui raport care prezintă lista clienților. Ca și formularele, rapoartele pot fi create utilizând aplicațiile wizard. Dacă este necesar, puteți intra în aplicația de proiectare pentru a face mici ajustări. Așa cum veți afla în curând, după crearea unui raport folosind o aplicație wizard, veți ști cum să creați toate rapoartele. Există și rapoarte complexe și, probabil, la un moment dat va fi necesar să creați și astfel de rapoarte. Pentru exemple de rapoarte complexe, studiați baza de date Northwind Traders, care este livrată împreună cu programul Access; o discuție detaliată despre rapoartele complexe depășește însă scopul cărții.

Este important să observați că munca mai dificilă constă în proiectarea interogărilor. Un raport nu este decât un mecanism de livrare a rezultatelor. Rapoartele bine proiectate nu fac decât un lucru: furnizează un format pentru afișarea eficientă a datelor.

Lista Client

Înainte de crearea raportului cu lista clienților, va fi necesară o interogare pe care să se bazeze raportul. În capitolul 8 am prezentat construirea unei interogări cu uniune. O astfel de interogare combină două declarații SQL în scopul creării unui set de rezultate. Întrucât clienții se împart în firme și persoane fizice, iar pentru stocarea numelor acestora sunt utilizate diferite câmpuri în funcție de tipul clientului, este necesară o interogare cu uniune. Figura 12.2 prezintă aplicația Query Designer și definiția interogării ClientListing.

FIGURA 12.2

Întrucât interogarea ClientListing utilizează clauza `Union`, interogarea poate fi creată numai folosind vizualizarea SQL.

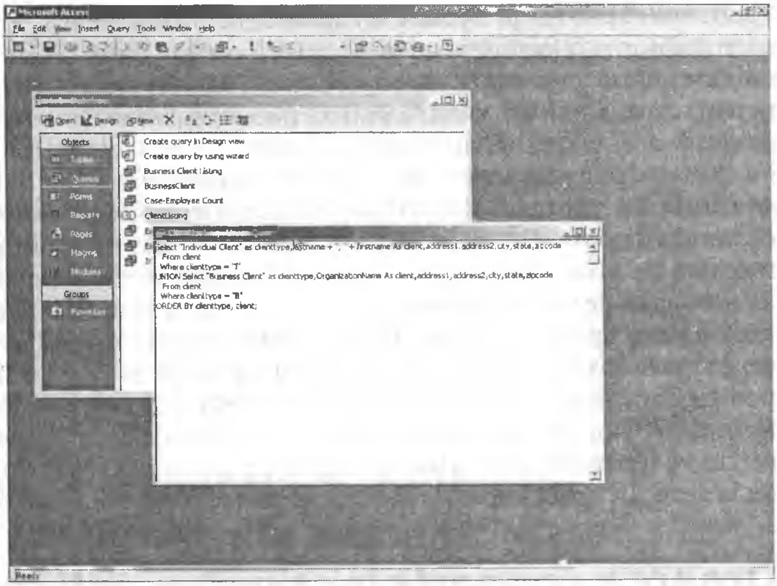


Figura 12.3 prezintă setul de rezultate ClientListing. Poate vă întrebați de ce sunt returnate în setul de rezultate cuvintele „Individual Client" și „Business Client", și nu codurile corespunzătoare tipurilor de clienți. Pentru utilizator, codul „I" sau „B" poate să nu fie la fel de intuitiv ca și cuvintele respective. Nu uitați, scopul unui raport este de a furniza informații. Utilizatorul nu trebuie să fie obligat să ghicească semnificația unei prescurtări.

FIGURA 12.3

Setul de rezultate al interogării ClientListing.

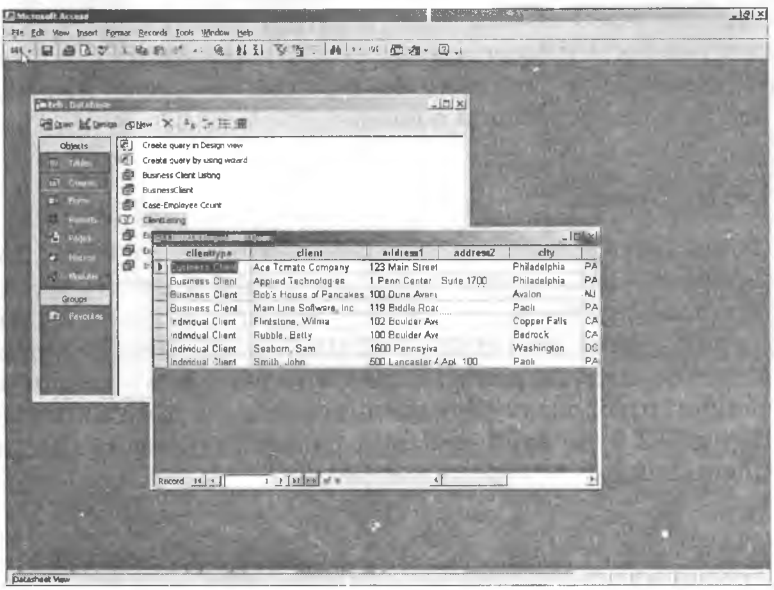
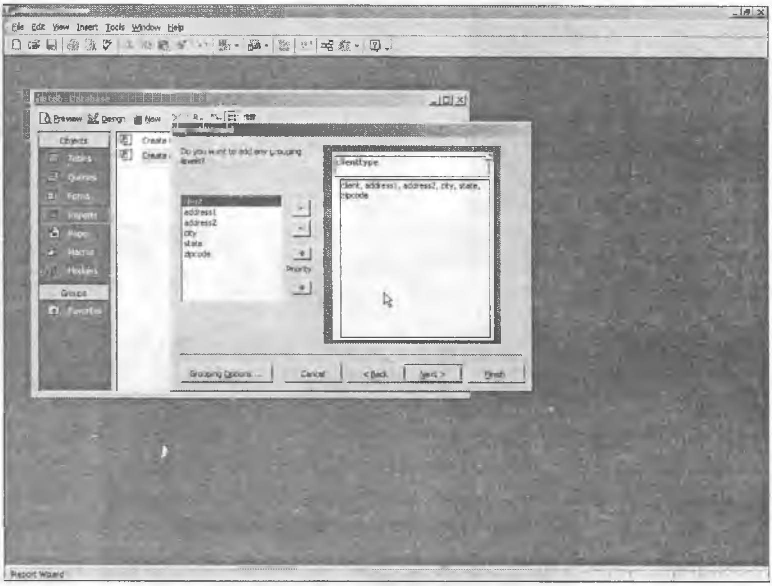


Figura 12.4 prezintă primul pas al aplicației Report Wizard. Ca și în cazul aplicației Form Wizard, primul pas solicită utilizatorului o interogare sau un tabel pe care să se bazeze raportul. La primul pas, puteți alege câmpurile pe care doriți să le afișați.

FIGURA 12.4

Pasul 1 al aplicației Report Wizard solicită utilizatorului o interogare sau un tabel pe care să se bazeze raportul.



După ce ați selectat câmpurile, puteți trece la pasul 2. Aici veți specifica modul de grupare a datelor în raport (vedeți figura 12.5). În mod implicit, un raport nu conține grupuri. În cazul listei clienților, datele vor fi grupate după tipul clienților. Dacă vă amintiți, trebuie să vă gândiți la această grupare când proiectați interogarea. De aceea, datele din interogarea ClientListing sunt ordonate mai întâi după tipul clienților.

Pasul 3, prezentat în figura 12.6, invită utilizatorul să specifice modul de sortare a datelor din raport. Dacă ați proiectat corect interogarea, nu este nevoie să specificați o altă ordine de sortare.

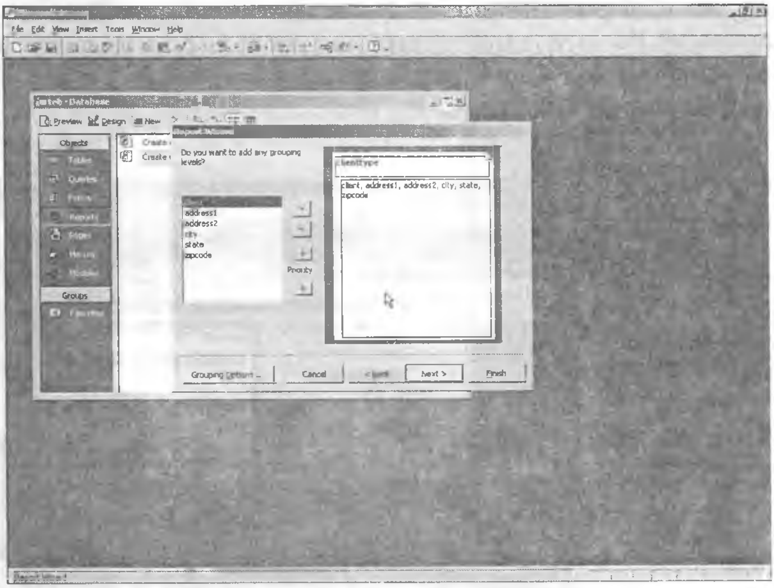


FIGURA 12.5

Pasul 2 al aplicației Report Wizard cere utilizatorului să precizeze modul de grupare a datelor din raport.

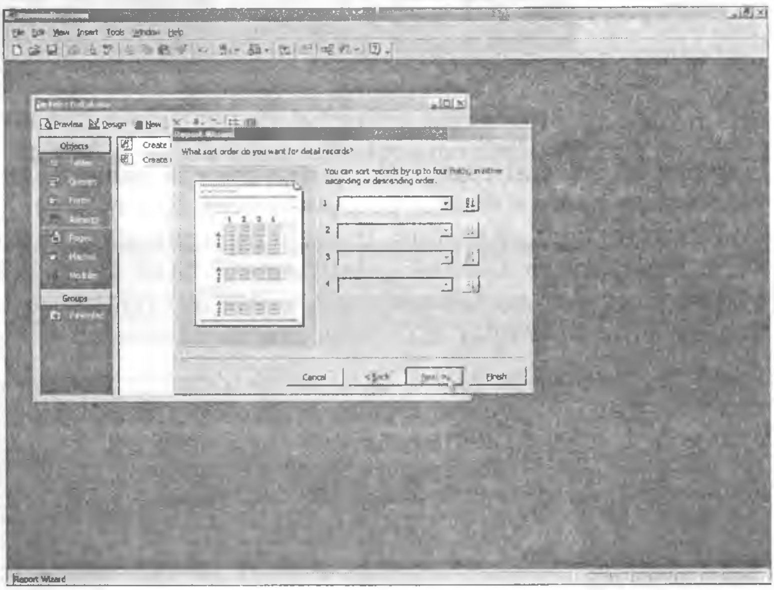


FIGURA 12.6

Pasul 3 al aplicației Report Wizard cere utilizatorului să precizeze modul de sortare a datelor din raport.

Pasul 4 al aplicației wizard cere utilizatorului să precizeze modul în care datele apar în raport, precum și orientarea paginii (vedeți figura 12.7).

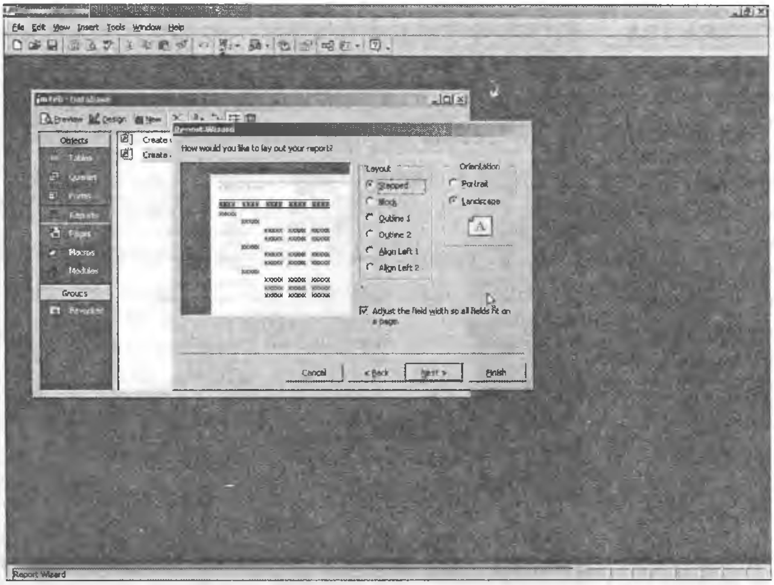


FIGURA 12.7
Pasul 4 al aplicației Report Wizard cere utilizatorului să specifice modul în care datele apar în raport și orientarea paginii.

Figura 12.8 prezintă pasul 5, care cere utilizatorului să specifice un stil al raportului. Access conține mai multe stiluri predefinite de raport.

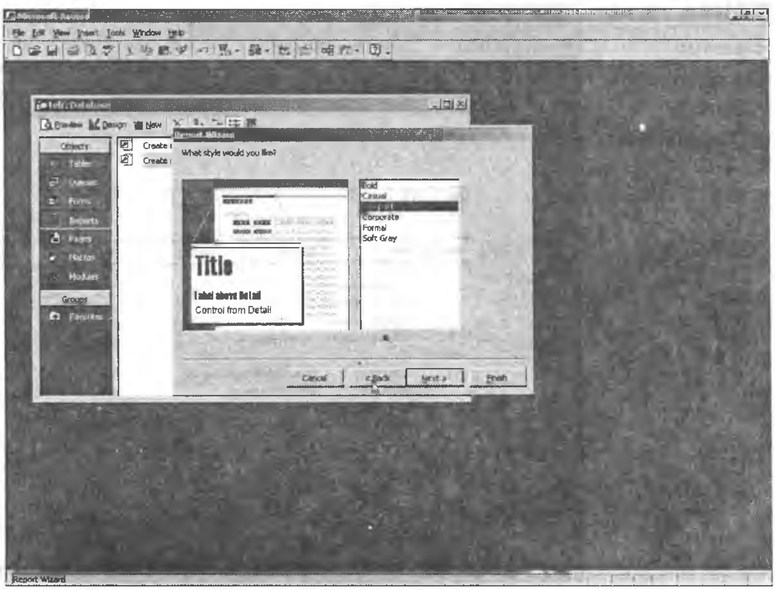


FIGURA 12.8
Pasul 5 al aplicației Report Wizard invită utilizatorul să specifice stilul raportului.

Ultimul pas al aplicației Report Wizard solicită utilizatorului un nume pentru raport și întreabă dacă raportul va fi previzualizat sau deschis în aplicația Report Designer (vedeți figura 12.9).

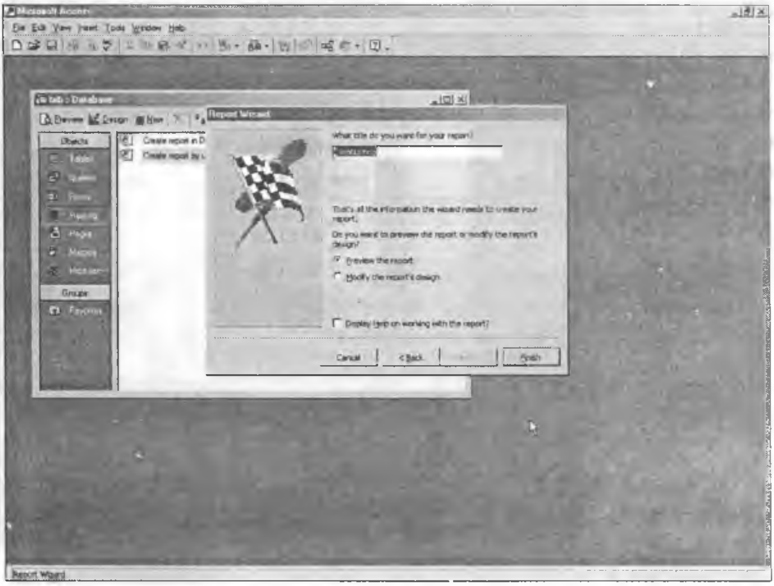


FIGURA 12.9
Ultimul pas al aplicației Report Wizard invită utilizatorul să specifice un nume pentru raport și îl întreabă dacă raportul va fi previzualizat sau modificat.

Figura 12.10 prezintă aspectul inițial al raportului. După cum vedeți, unele date sunt trunchiate. Secțiunea următoare vă arată cum puteți modifica designul raportului.

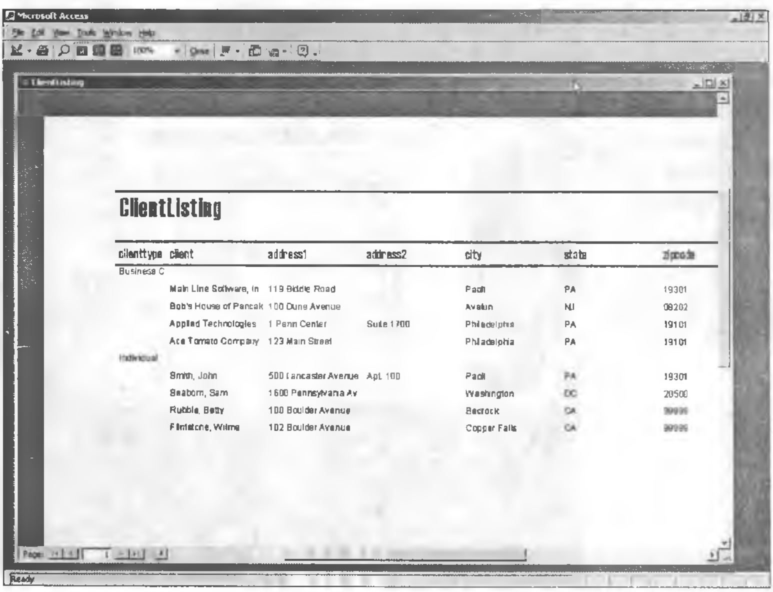


FIGURA 12.10
Raportul ClientListing produs de aplicația Report Wizard.

Modificarea raportului ClientListing în Report Designer

Modificarea rapoartelor este asemănătoare modificării formularelor. Când executați clic pe fiecare element din aplicația de proiectare a rapoartelor, puteți manevra diferitele atribute ale elementului selectat în fereastra Property. Alternativ, puteți lucra direct cu obiectul pentru a-i modifica dimensiunile și/sau poziția. Figura 12.11 prezintă raportul modificat în aplicația de proiectare.

FIGURA 12.11

Rapoartele pot fi modificate la fel ca și formularele.

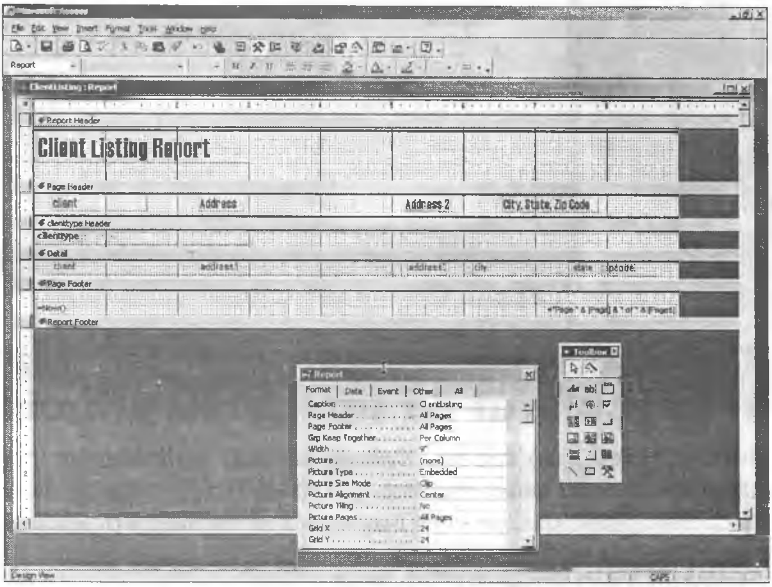


Figura 12.12 prezintă aspectul modificat al raportului ClientListing în modul Print Preview.

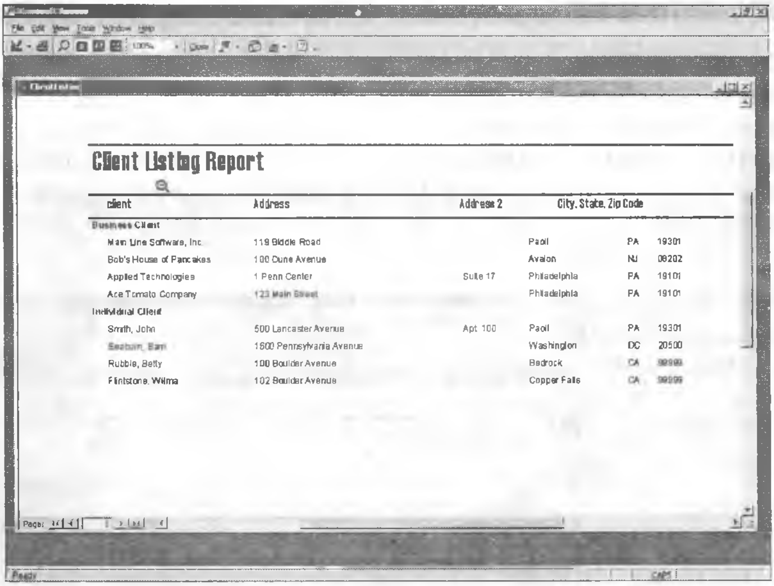


FIGURA 12.12
Raportul modificat ClientListing în modul Print Preview.

Introducerea elementelor din raport în formularul de tip meniu principal

După ce creați un raport, trebuie să dați utilizatorilor posibilitatea de a rula raportul. În acest context, puteți oferi acces la raport în formularul de tip meniu principal, pe care l-ați creat în capitolul precedent. Figura 12.13 prezintă primul pas pentru introducerea unui buton al raportului în formularul de tip meniu principal.

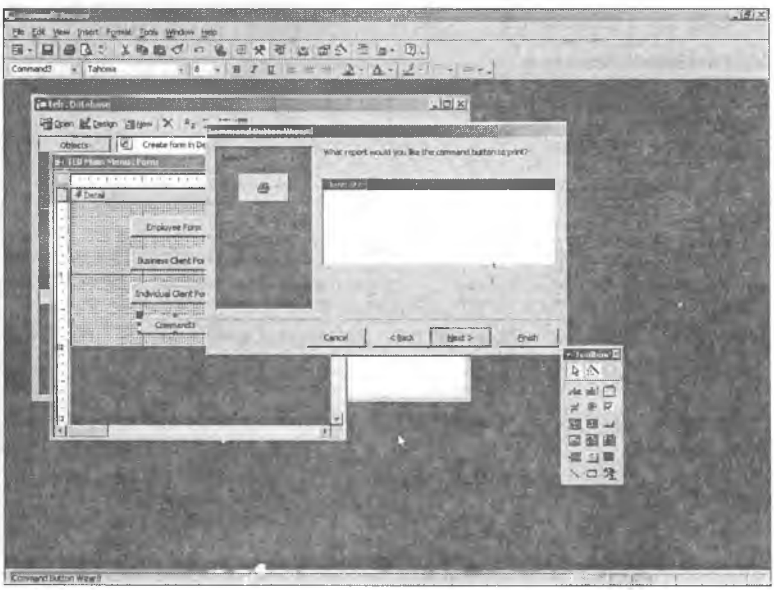


FIGURA 12.13
Opțiunile pentru butoane ale rapoartelor se găsesc în secțiunea Report Operations a aplicației Command Button Wizard.

Pasul 2 solicită numele raportului care va fi rulat la apăsarea butonului (vedeți figura 12.14).

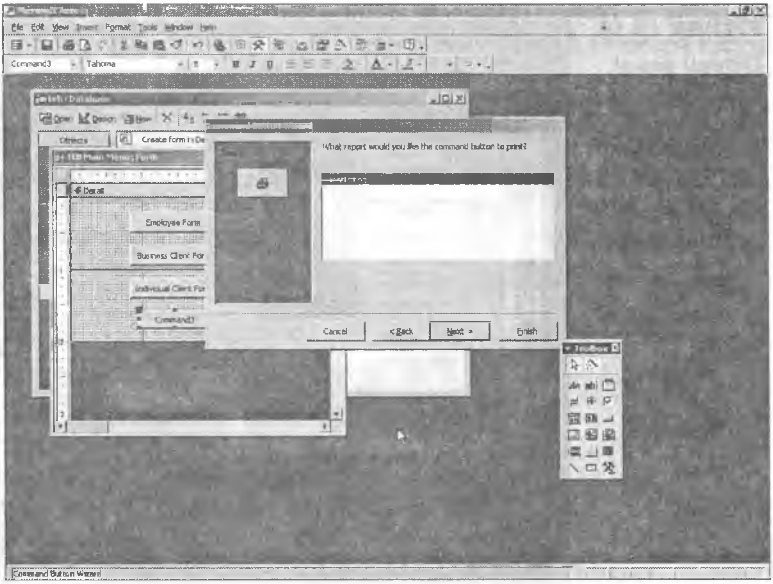


FIGURA 12.14
Pasul 2 al aplicației Command Button Wizard vă solicită numele raportului care va fi rulat la apăsarea butonului.

Pasul 3, prezentat în figura 12.15, vă solicită să precizați dacă pe buton va apărea un text sau o imagine.

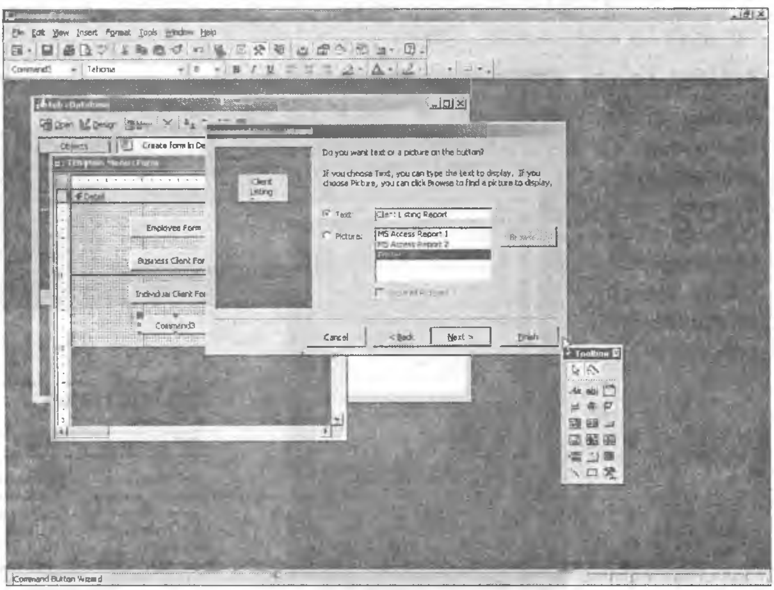
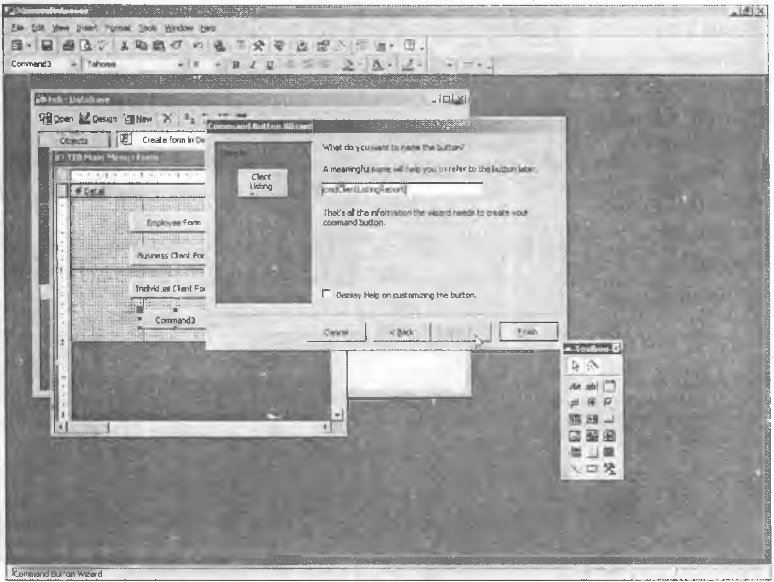


FIGURA 12.15
Pasul 3 vă solicită să precizați dacă pe buton va apărea un text sau o imagine.

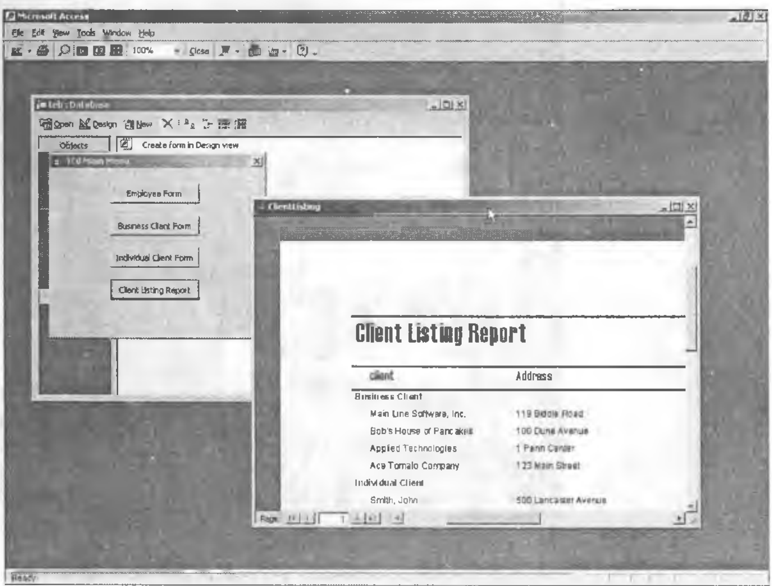
Ultimul pas al aplicației wizard invită utilizatorul să furnizeze un nume descriptiv pentru buton (vedeți figura 12.16).

FIGURA 12.16
Ultimul pas al aplicației Command Button Wizard solicită un nume descriptiv pentru buton.



În final, figura 12.17 vă prezintă formularul de tip meniu principal care lansează raportul ClientListing în modul Preview. Dacă rulați din nou aplicația Command Button Wizard, vi se va părea foarte simplu să introduceți un buton al cărui efect este printarea rezultatelor raportului.

FIGURA 12.17
Formularul de tip meniu principal are acum și capacitatea de a lansa raportul ClientListing în modul Print Preview.



Rezumat

Crearea rapoartelor, ca și a formularelor, este o sarcină simplă în Access. Utilizând aplicațiile wizard, nu sunteți obligat să scrieți coduri de program pentru a produce rezultate valoroase și utile. Gândindu-vă la acest lucru, nu încape îndoială că dacă știți să scrieți coduri de program aveți o capacitate mult mai mare de a crea componente mai flexibile și mai sofisticate pentru aplicații. Acum după ce ați parcurs această carte, următorul pas logic este să explorați și alte funcții din Access și să învățați să programați în Visual Basic. Indiferent ce veți face, nu uitați că totul pornește de la o bază de date! Succes!

A



STRUCTURA BAZEI DE DATE TEB

Tabelul Case

Numele câmpului	Tipul de date	Descriere
CaseID	AutoNumber	Cheie primară
DocketNumber	Text	Numărul de rol
Notes	Memo	Câmp pentru note în format liber
StartDate	Date/Time	Data la care începe lucrul la caz
TrialDate	Date/Time	Data procesului
SettlementDate	Date/Time	Data înțelegerii
DepartmentID	Number	Cheie externă la tabelul Department
CourtID	Number	Cheie externă la tabelul Court
JudgeID	Number	Cheie externă la tabelul Judge

Tabelul Client

Numele câmpului	Tipul de date	Descriere
ClientID	AutoNumber	Cheie primară
ClientType	Text	Tipul clientului (B pentru firme, I pentru persoane fizice)
FirstName	Text	Prenumele (dacă ClientType = I)
LastName	Text	Numele de familie (dacă ClientType = I)
MiddleInitial	Text	Inițiala (dacă ClientType = I)
OrganizationName	Text	Numele firmei (dacă ClientType = B)
Address1	Text	Prima adresă
Address2	Text	A doua adresă
City	Text	Orașul
State	Text	Statul
ZipCode	Text	Codul poștal
Phone	Text	Numărul de telefon
Email	Text	Adresa de e-mail

Tabelul ClientCase

Numele câmpului	Tipul de date	Descriere
ClientCaseID	AutoNumber	Cheie primară
ClientID	Number	Cheie externă la tabelul Client
CaseID	Number	Cheie externă la tabelul Case

Tabelul Contact

Numele câmpului	Tipul de date	Descriere
ContactID	AutoNumber	Cheie primară
FirstName	Text	Prenumele
MiddleInitial	Text	Inițiala
LastName	Text	Numele de familie
Title	Text	Funcția
Phone	Text	Numărul de telefon
Extension	Text	Interiorul
Email	Text	Adresa de e-mail
ClientID	Number	Cheie externă la tabelul Client

Tabelul Court

Numele câmpului	Tipul de date	Descriere
CourtID	AutoNumber	Cheie primară
Description	Text	Numele instanței

Tabelul Department

Numele câmpului	Tipul de date	Descriere
DepartmentID	AutoNumber	Cheie primară
Description	Text	Numele departamentului

Tabelul Employee

Numele câmpului	Tipul de date	Descriere
EmployeeID	AutoNumber	Cheie primară
FirstName	Text	Prenumele
MiddleInitial	Text	Inițiala
LastName	Text	Numele de familie
SocialSecurity	Text	Numărul asigurării sociale
Address1	Text	Prima adresă
Address2	Text	A doua adresă
City	Text	Orașul
State	Text	Statul
ZipCode	Text	Codul poștal

HomePhone	Text	Numărul de telefon de acasă
WorkExtension	Text	Interiorul de la serviciu
Email	Text	Adresa de email
DepartmentID	Number	Cheie externă la tabelul Department
EmployeeClassID	Number	Cheie externă la tabelul EmployeeClass

Tabelul EmployeeCase

Numele câmpului	Tipul de date	Descriere
EmployeeCaseID	AutoNumber	Cheie primară
CaseID	Number	Cheie externă la tabelul Case
EmployeeID	Number	Cheie externă la tabelul Employee

Tabelul EmployeeClass

Numele câmpului	Tipul de date	Descriere
EmployeeClassID	AutoNumber	Cheie primară
Description	Text	Numele clasei de angajați
Default	Currency	Tariful implicit

Tabelul Invoice

Numele câmpului	Tipul de date	Descriere
InvoiceID	AutoNumber	Cheie primară
InvoiceNumber	Text	Numărul facturii
InvoiceRate	Date/Time	Data facturii
ClientID	Number	Cheie externă la tabelul Client

Tabelul Judge

Numele câmpului	Tipul de date	Descriere
JudgeID	AutoNumber	Cheie primară
FirstName	Text	Prenumele
MiddleInitial	Text	Inițiala

LastName	Text	Numele de familie
Notes	Memo	Note ale utilizatorului
CourtID	Number	Cheie externă la tabelul Court

Tabelul TimeEntryDetail

Numele câmpului	Tipul de date	Descriere
TimeEntryDetail	AutoNumber	Cheie primară
Description	Text	Descrierea activității
Hours	Number	Numărul de ore alocate
Rate	Currency	Tariful pe oră
WorkDate	Date/Time	Data activității
WorkCategory	Number	Cheie externă la tabelul WorkCategory
InvoiceID	Number	Cheie externă la tabelul Invoice
CaseID	Number	Cheie externă la tabelul Case
EmployeeID	Number	Cheie externă la tabelul Employee
ClientID	Number	Cheie externă la tabelul Client

Tabelul WorkCategory

Numele câmpului	Tipul de date	Descriere
WorkCategoryID	AutoNumber	Cheie primară
Description	Text	Descrierea categoriei de activități
DefaultRate	Currency	Tariful implicit pe oră

B



REGULI DE INTEGRITATE REFERENȚIALĂ ÎN BAZA DE DATE TEB

Toate relațiile implementează ștergeri restricționate și introducerea restricționată a datelor. Actualizările restricționate nu sunt aplicabile deoarece sunt utilizate chei primare surogat. Paragrafele următoare prezintă fiecare tabel părinte și tabelele copil corespunzătoare lui.

Tabelul Case (cheie primară: `caseid`)

Tabelul Employee: mulți la mulți (prin tabelul de legătură EmployeeCase);

Tabelul Client: mulți la mulți (prin tabelul de legătură ClientCase);

Tabelul TimeEntryDetail: unu la mulți.

Tabelul Client (cheie primară: `clientid`)

Tabelul ClientCase: unu la mulți (facilitează relația mulți la mulți a tabelului Case);

Tabelul Contact: unu la mulți;

Tabelul Invoice: unu la mulți;

Tabelul TimeEntryDetail: unu la mulți.

Tabelul Court (cheie primară: `CourtID`)

Tabelul Case: unu la mulți;

Tabelul Judge: unu la mulți.

Tabelul Department (cheie primară: `departmentid`)

Tabelul Case: unu la mulți;

Tabelul Employee: unu la mulți.

Tabelul Employee (cheie primară: `employeeid`)

Tabelul EmployeeCase: unu la mulți (facilitează relația mulți la mulți a tabelului Case);

Tabelul TimeEntryDetail: unu la mulți.

Tabelul EmployeeClass

Tabelul Employee: unu la mulți.

Tabelul Invoice (cheie primară: `InvoiceID`)

Tabelul TimeEntryDetail: unu la mulți.

C



GLOSAR

A doua formă normală

A doua formă normală izolează și mută datele redundante într-un tabel separat.

A treia formă normală

A treia formă normală face ca toate câmpurile unui rând să fie dependente de cheia primară.

A patra formă normală

A patra formă normală izolează relațiile multiple independente. Este foarte rar utilizată.

A cincea formă normală

A cincea formă normală izolează relațiile multiple înrudite semantic. Este foarte rar utilizată.

Administrator al bazei de date

Administratorul bazei de date răspunde de administrarea și organizarea unui sistem de administrare a bazelor de date.

Alias

Un alias este o variație a numelui unui tabel într-o interogare SQL. De exemplu, interogarea următoare utilizează aliasul c în locul numelui real al tabelului Customers:

```
SELECT C.CompanyName, *
     FROM Customers c;
```

Atribut

- Un câmp/o coloană dintr-un tabel.
- O proprietate a unui câmp, ca de exemplu legenda, tipul de date, dimensiunea ș.a.m.d.

Auto-uniune

Uniune în care un tabel este unit cu el însuși. Un exemplu tipic este o relație între tabelele cu date despre angajați (Employee) și superiorii lor ierarhici (Supervisor). Toți superiorii sunt angajați. Fiecărui angajat îi corespunde un câmp SupervisorID, care referă alt câmp EmployeeID.

Bază de date

O colecție de date administrate și organizate de un sistem de administrare a bazelor de date.

Blocaj (deadlock, deadly embrace)

Situație în care două sau mai multe procese nu pot continua deoarece au nevoie de aceeași resursă.

Blocare

Un dispozitiv de control utilizat pentru a oferi acces exclusiv la o anumită resursă, de obicei o înregistrare.

Câmp

Câmpurile, numite și *coloane*, sunt elementele de construcție ale unui tabel. De exemplu, un tabel tipic cu date despre clienți este compus din câmpuri pentru prenume, numele de familie, oraș, stat și codul poștal. Un câmp (numit și *atribut*) este o coloană a unui tabel și stochează un element atomic descriptiv despre o entitate.

Cheie

Un câmp utilizat pentru accesul la o anumită înregistrare. Printre tipurile de chei se numără cheile primare și cheile candidat.

Cheie candidat

Un câmp sau un grup de câmpuri care identifică unic o înregistrare.

Cheie compusă

Un tip de cheie primară alcătuită din două sau mai multe câmpuri dintr-un tabel.

Cheie externă (străină)

O cheie stocată în alt tabel cu scopul de a suporta o relație. De exemplu, cheia primară a tabelului pentru comenzi este preluată ca și cheie externă în tabelul cu date despre comenzi.

Cheie primară

Un câmp sau o combinație de câmpuri care identifică în mod unic un rând de date. Există două tipuri de chei primare: chei surogat și chei simple.

Cheie surogat

O cheie surogat este un câmp creat pentru identificarea unică a unui rând de date. De obicei, o cheie surogat este de tipul întreg cu autoincrementare. Celălalt tip de cheie (cheie primară) reprezintă o cheie compusă, alcătuită din două sau mai multe câmpuri. Alternativa la o cheie surogat este o cheie naturală (care are sens în afară de faptul că este cheie primară). Cheile naturale nu sunt neapărat compuse, cum este cazul numărului clientului, numărul facturii sau numărul comenzii, toate acestea fiind chei naturale, alcătuite dintr-un singur câmp.

Concurență

Acces simultan la aceleași date al mai multor utilizatori.

Conflict

Conflictul resurselor are loc dacă doi sau mai mulți utilizatori încearcă să actualizeze simultan același element de date.

Copil

Ținta unei relații. De exemplu, într-o relație între comenzi și elementele comenzii, elementele comenzii sunt copii, deoarece sunt ținta unei relații cu comenzile.

Data Definition Language (DDL)

Sublimbajul DDL permite utilizatorilor să creeze, să modifice și să șteargă diferite elemente ale unei baze de date, ca tabele, coloane și vizualizări.

Data Manipulation Language (DML)

Sublimbajul DML permite utilizatorilor să creeze, să actualizeze, să șteargă și să interogheze elementele de date dintr-o bază de date.

Date

Conținutul unei baze de date. Elementele de date dintr-o bază de date pot fi extrase, grupate și sumarizate pentru a produce informații.

Date care se repetă

Valoare a unui câmp care apare de mai multe ori într-un tabel. De exemplu, descrierea unui tip de client poate apărea de mai multe ori. Procesul de normalizare va muta o singură instanță a fiecărei descrieri într-un tabel separat pentru a elimina datele care se repetă și redundanțele asociate lor.

Date redundante

Date stocate în cel puțin două locuri într-o bază de date. Scopul normalizării este eliminarea datelor redundante.

Denormalizare

Procesul invers normalizării.

Design conceptual al bazei de date

Un model conceptual al bazei de date, independent de implementarea unei anumite baze de date.

Designul (proiectarea) bazei de date

Procesul de identificare a entităților din realitate și a relațiilor dintre ele, urmat de transformarea acestora într-un model al bazei de date.

Diagrama relațiilor între entități (Entity Relationship Diagram – ERD)

O tehnică specială de realizare a diagramelor care permite prezentarea proiectului bazei de date relaționale.

Dicționar de date

Datele care descriu structura unei baze de date; denumite și *metadate* – date care descriu datele. De exemplu, tipul și dimensiunea datelor sunt metadate care descriu un anumit câmp de date. Colecția acestor date formează dicționarul de date al unei baze de date.

Domeniu

Setul valorilor posibile pe care le poate conține un element de date. De exemplu, un câmp care conține ziua din lună poate avea valori de la 1 la 31.

Entitate

Un obiect al bazei de date capabil să stocheze datele; mai este numit și *tabel*. Un obiect al bazei de date care stochează date despre o persoană, un loc, un lucru sau un concept și este organizat sub formă de rânduri (înregistrări) și coloane (câmpuri).

Forma normală Boyce-Codd

Forma normală Boyce-Codd este o extensie a celei de a treia forme normale. Un tabel se află în forma normală Boyce-Codd dacă dependența fiecărui element de date dintr-un tabel poate fi urmărită cu o cheie candidat. Pentru exemplificare, să considerăm structura următoare: Student, Profesor și Decan. Profesor depinde de Decan, dar Profesor nu este cheie. Pentru a trece la forma normală Boyce-Codd, structura trebuie divizată în două: Student, Profesor și Profesor, Decan.

Funcții de agregare

Funcțiile de agregare din SQL, cum sunt `Count()`, `Sum()`, `Avg()`, `Min()` și `Max()` oferă posibilitatea de a sumariza datele la efectuarea unei interogări.

Independența datelor

O separație între modul de stocare a datelor și modul de accesare și manipulare a acestora.

Index

Un element al bazei de date, utilizat pentru a optimiza anumite căi de acces la date. De exemplu, dacă trebuie să accesați date despre clienți pornind de la oraș, trebuie să realizați un index pe baza câmpului cu date despre oraș.

Integritate referențială

Restricții care susțin relațiile dintre tabele. De exemplu, în cazul clienților și al comenzilor, un utilizator va fi împiedicat să șteargă înregistrarea unui client dacă există înregistrări înrudite cu comezi date de el. Mai mult, un utilizator va fi împiedicat să creeze o înregistrare a unei comenzi pentru un părinte (în acest caz, o înregistrare pentru un client corespunzător) inexistent.

Interogare

Procesul de extragere a datelor dintr-o bază de date folosind comezi a căror sintaxă poate fi înțeleasă de sistemul de administrare a bazelor de date. Cea mai obișnuită cale de interogare a datelor este utilizarea comenzii Select din SQL.

Înregistrare

Un grup de câmpuri care fac parte din același tabel. Înregistrările mai sunt numite și *rânduri* sau *tupluri*. O înregistrare este un rând al unui tabel și conține toate câmpurile pentru o anumită instanțiere a unei entități.

Metadate

Date care descriu alte date. Datele conținute în dicționarul de date (Data Dictionary) sunt adesea denumite metadate care descriu structura bazei de date.

Modelul ierarhic al bazei de date

Un model al bazei de date în care datele sunt stocate ierarhic. Pentru a accesa datele, trebuie să începeți din partea superioară a ierarhiei și să pătrundeți până la elementul de date dorit. XML este un exemplu de stocare ierarhică a datelor.

Modelul relațional al bazei de date

Un model al bazei de date alcătuit din tabele bidimensionale înrudite între ele. E. F. Codd a dezvoltat modelul relațional al bazei de date în anul 1970.

Normalizare

Procesul de organizare a unei baze de date astfel încât să fie eliminate datele redundante.

Ordinea fizică a înregistrărilor

Ordinea înregistrărilor, așa cum sunt ele stocate pe hard disc.

Ordonare logică a înregistrărilor

O ordonare a înregistrărilor bazată pe un anumit index.

Părinte

Sursa unei relații. De exemplu, într-o relație între comenzi și elemente ale comenzilor, tabelul pentru comenzi este părinte, deoarece este sursa unei relații cu tabelul pentru elemetele comenzilor.

Prima formă normală

Prima formă normală însemnă eliminarea grupurilor care se repetă.

Proiectul (designul) bazei de date fizice

Rezulatul implementării proiectului conceptual al bazei de date într-un anumit sistem de administrare a bazelor de date.

Rând

Un grup de câmpuri care fac parte din același tabel. Mai sunt denumite *înregistrări* sau *tupluri*.

Relație mulți la mulți

O relație între două tabele, în care mai multe înregistrări dintr-un tabel pot fi înrudite cu mai multe înregistrări din celălalt tabel. Pentru a facilita o relație mulți la mulți, trebuie utilizat un tabel intermediar de legătură.

Relație unu la mulți

O relație între două tabele în care o înregistrare din tabelul părinte este înrudită cu mai multe înregistrări din tabelul copil. De exemplu, o înregistrare a unui client (din tabelul părinte) poate fi înrudită cu mai multe înregistrări ale comenzilor (în tabelul copil).

Restricții de domeniu

Restricții asupra unui element de date, impuse pentru ca numai anumite valori din domeniu să poată fi stocate. Mai sunt numite și *reguli de validare la nivel de coloană/câmp.*

Sistem de administrare a bazelor de date (Database Management System – DBMS)

Software cu capacitatea de definire și administrare a bazelor de date. Printre sistemele populare de administrare a bazelor de date sunt: SQL Server, Oracle și DB2.

Structured Query Language (SQL)

SQL este un limbaj independent pentru baze de date, utilizat pentru a inteorga, actualiza, șterge și modifica datele.

Tabel

Un tabel este o reprezentare a unei entități și este alcătuit din rânduri și coloane de date.

Tabel de legătură

Un tabel intermediar utilizat pentru a suporta o relație mulți la mulți.

Tranzacție

Un set de operații care trebuie fie să reușească integral fie să eșueze. De exemplu, o operație care adaugă un nou client și o nouă comandă trebuie să reușească integral sau să eșueze. Dacă ar fi posibilă adăugarea unui client fără introducerea unei comenzi și/sau adăugarea unei comenzi fără adăugarea unui client, ar fi apărea o violare a integrității bazei de date.

Tuplu

Un grup de câmpuri care fac parte din același tabel. Tuplurile mai sunt numite și *rânduri* sau *înregistrări.*

Uniune

O operație în SQL care combină datele din două tabele într-un singur set de rezultate. O uniune combină numai două tabele; o interogare care necesită combinarea mai multor tabele utilizează uniuni multiple (fiecare dintre ele combinând două tabele).

Uniune

O operațiune în SQL care unește două declarații Select pentru a obține un singur set de rezultate.

Valoare nulă (null value)

Indică situația în care un utilizator nu a atribuit o valoare unui element de date. Valoarea nulă este diferită de 0 sau de un șir vid. De exemplu, dacă înregistrarea despre un client conține o valoare nulă în câmpul pentru stat, aceasta înseamnă că statul nu este cunscut. Gândiți-vă la valoarea nulă ca la o completare a câmpului cu textul „Nu știu".

Vizualizare

Un obiect al bazei de date care reprezintă o colecție de date din unul sau mai multe tabele. Mai este numită și *interogare*.

Wildcard

Un operator wildcard permite utilizatorului să găsească date pe baza unui model. De exemplu, numele clienților pot fi comparate cu A* pentru a găsi toate înregistrările cu clienții al căror nume începe cu litera A.



D

PERIODICE DE SPECIALITATE

Iată câteva reviste informative populare, care au ca subiect dezvoltarea bazelor de date:

- SQL Server Magazine (www.sqlmag.com);
- Oracle Magazine (www.oramag.com);
- DB2 Magazine (www.db2.mag);
- Sybase Magazine (www.sybase.com/inc/sybmag);
- Access/VB/SQL Advisor (www.advisor.com/AccessVBSQLAdvisor);
- Visual Studio Magazine (www.devx.com);
- Intelligent Enterprise (www.intelligententerprise.com);
- DM Review (www.dmreview.com);
- Inside Microsoft Access (www.elementkjournals.com);
- Smart Access (www.pinpub.com/access);
- Database Administrator Newsletter (www.tdan.com);
- Database Trends (www.databasetrends.com).

E



RESURSE PE WEB

Iată câteva site-uri web populare. a căror temă este dezvoltarea bazelor de date:

- `www.sqlservercentral.com` – O comunitate de dezvoltatori SQL Server și administratori de baze de date.
- `www.devx.com` – Devx este un site web informativ pentru dezvoltarea aplicațiilor, care este proprietate a Fawcette Publications.
- `www.msdn.microsoft.com/newsgroups` – Oferă acces la grupurile de discuții online pentru suport peer-to-peer.
- `www.sqlmag.com` – Varianta online a SQL Server Magazine.
- `www.microsoft.com/office/access/` - Site-ul web Microsoft Access.
- `www.databaseanswers.com` – Un site informativ general, drăguț, despre analiza și proiectarea bazelor de date, precum și despre cele mai bune metode.